



# **NAVAL POSTGRADUATE SCHOOL**

**MONTEREY, CALIFORNIA**

## **THESIS**

**AN ANALYSIS OF RELATED SOFTWARE CYCLES AMONG  
ORGANIZATIONS, PEOPLE AND THE SOFTWARE  
INDUSTRY**

by

Robert Moore  
Brady Adams

June 2008

Thesis Advisor:  
Second Reader:

Glenn Cook  
Thomas Housel

**Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

<b>REPORT DOCUMENTATION PAGE</b>			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
<b>1. AGENCY USE ONLY (Leave blank)</b>		<b>2. REPORT DATE</b> June 2008	<b>3. REPORT TYPE AND DATES COVERED</b> Master's Thesis	
<b>4. TITLE AND SUBTITLE</b> An Analysis of Related Software Cycles Among Organizations, People and the Software Industry			<b>5. FUNDING NUMBERS</b>	
<b>6. AUTHOR(S)</b> LT Robert Moore, CPT Brady Adams				
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Naval Postgraduate School Monterey, CA 93943-5000			<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>	
<b>9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> N/A			<b>10. SPONSORING/MONITORING AGENCY REPORT NUMBER</b>	
<b>11. SUPPLEMENTARY NOTES</b> The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
<b>12a. DISTRIBUTION / AVAILABILITY STATEMENT</b> Approved for public release; distribution is unlimited			<b>12b. DISTRIBUTION CODE</b>	
<b>13. ABSTRACT (maximum 200 words)</b>  <p>There is a need to understand cycles associated with software upgrades as they effect people, organizations and the software industry. This thesis intends to explore the moderating factors of these three distinct and disjointed cycles and propose courses of action towards mitigating various issues and problems inherent in the software upgrade process.</p> <p>This thesis will acknowledge that three related but disjointed cycles are common in many software upgrade ventures in today's organizations:</p> <ul style="list-style-type: none"> <li>a. End-user characteristics in adapting to new software</li> <li>b. Organizational ability to adopt new software</li> <li>c. The software industry's motivation and processes in introducing new software</li> </ul> <p>Realizing the importance of these related cycles involves developing an understanding of several aspects we research in this study. First, awareness in understanding why users adopt new software and the demographic factors involved, such as gender, age and experience are considered. Second, we present how organizations integrate new software by exploring factors such as cost, time, reliability and benefit analysis. Last, we provide evidence supporting motivating forces and factors behind software introduction rates within the software industry. These important aspects together culminate in cyclical phenomenon managers and executives need to be aware of, as implementing new software upgrades have become an inevitable undertaking in most of today's organizations.</p>				
<b>14. SUBJECT TERMS</b> Software implementation, Software cycles, Software in the DoD			<b>15. NUMBER OF PAGES</b> 89	
			<b>16. PRICE CODE</b>	
<b>17. SECURITY CLASSIFICATION OF REPORT</b> Unclassified	<b>18. SECURITY CLASSIFICATION OF THIS PAGE</b> Unclassified	<b>19. SECURITY CLASSIFICATION OF ABSTRACT</b> Unclassified	<b>20. LIMITATION OF ABSTRACT</b> UU	

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited**

**AN ANALYSIS OF RELATED SOFTWARE CYCLES AMONG ORGANIZATIONS,  
PEOPLE AND THE SOFTWARE INDUSTRY**

Robert W. Moore  
Lieutenant, United States Navy  
B.S., Troy State University, 1999

Submitted in partial fulfillment of the  
requirements for the degree of

**MASTER OF SCIENCE IN INFORMATION SYSTEMS AND TECHNOLOGY**

from the

**NAVAL POSTGRADUATE SCHOOL  
June 2008**

Brady Adams  
Captain, United States Army  
B.S., Excelsior College, 2005

Submitted in partial fulfillment of the  
requirements for the degree of

**MASTER OF SCIENCE IN INFORMATION SYSTEMS AND TECHNOLOGY**

from the

**NAVAL POSTGRADUATE SCHOOL  
September 2008**

Author: LT Robert W. Moore

CPT Brady Adams

Approved by: Mr. Glenn R. Cook  
Thesis Advisor

Dr. Thomas J. Housel  
Second Reader

Dr. Dan Boger  
Chairman, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

## **ABSTRACT**

There is a need to understand cycles associated with software upgrades as they affect people, organizations and the software industry. This thesis intends to explore the moderating factors of these three distinct and disjointed cycles and propose courses of action towards mitigating various issues and problems inherent in the software upgrade process.

This thesis will acknowledge that three related but disjointed cycles are common in many software upgrade ventures in today's organizations:

- d. End-user characteristics in adapting to new software
- e. Organizational ability to adopt new software
- f. The software industry's motivation and processes in introducing new software

Realizing the importance of these related cycles involves developing an understanding of several aspects we research in this study. First, awareness in understanding why users adopt new software and the demographic factors involved, such as gender, age and experience are considered. Second, we present how organizations integrate new software by exploring factors such as cost, time, reliability and benefit analysis. Last, we provide evidence supporting motivating forces and factors behind software introduction rates within the software industry. These important aspects together culminate in cyclical phenomenon managers and executives need to be aware of, as implementing new software upgrades have become an inevitable undertaking in most of today's organizations.

THIS PAGE INTENTIONALLY LEFT BLANK

## TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	BACKGROUND .....	1
1.	Known Upgrade Problems .....	5
2.	Importance of Upgrade Issues .....	8
II.	METHODOLOGY.....	11
A.	QUALITATIVE APPROACH .....	11
B.	RESEARCH DESIGN.....	12
C.	CONCEPTUAL DESIGN .....	13
1.	Description of the Diagram .....	14
III.	USERS, ORGANIZATIONS AND SOFTWARE INDUSTRY .....	15
A.	USER ADAPTATION CYCLES .....	15
1.	Exploring the UTAUT Model .....	15
a.	<i>Performance Expectancy</i> .....	19
b.	<i>Effort Expectancy</i> .....	21
c.	<i>Social Influence</i> .....	22
d.	<i>Facilitating Conditions</i> .....	24
2.	Exploring a Revised UTAUT Model .....	26
a.	<i>Training</i> .....	28
b.	<i>Shared Belief</i> .....	29
c.	<i>Communication</i> .....	29
B.	ORGANIZATION ADOPTION CYCLES .....	31
1.	Critical Software Upgrade Questions .....	32
a.	<i>Motivating Forces</i> .....	34
b.	<i>Contingency Forces</i> .....	36
c.	<i>Decision</i> .....	36
d.	<i>Impacts</i> .....	36
e.	<i>Corrective Actions</i> .....	36
f.	<i>Planned Strategies</i> .....	37
2.	Software Implementation .....	37
3.	Case Study on Windows Vista Implementation .....	42
a.	<i>Future Preparedness</i> .....	44
C.	SOFTWARE INDUSTRY DEVELOPMENT CYCLES .....	45
1.	Vendor Benefits from Improving Development Times .....	47
2.	Money, Time and Quality.....	47
3.	Lessons for the Software Industry .....	50
a.	<i>Lessons for Practice</i> .....	51
b.	<i>Lessons for Cycle Time Theory</i> .....	53
IV.	CONCLUSIONS AND RECOMMENDATIONS.....	57
A.	ANALYSIS AND RECOMMENDATIONS .....	57
B.	TIMING AND FLEXIBILITY .....	62

C.	STARTING POINT IN USING THIS RESEARCH METHOD.....	63
D.	FUTURE DIRECTIONS CONCERNING THE USER CYCLE .....	66
E.	FUTURE DIRECTIONS CONCERNING THE SOFTWARE INDUSTRY CYCLE .....	67
F.	FUTURE DIRECTIONS CONCERNING THE ORGANIZATION CYCLE .....	67
	LIST OF REFERENCES.....	69
	INITIAL DISTRIBUTION LIST .....	73

## LIST OF FIGURES

Figure 1.	Windows Software Cycle From Windows History <a href="http://www.microsoft.com/windows/WinHistoryProGraphic.mspix">http://www.microsoft.com/windows/WinHistoryProGraphic.mspix</a> <u>Last accessed May 2008</u> .....	4
Figure 2.	Software Upgrade Cycle.....	13
Figure 3.	UTAUT Research Model From (Venkatesh, V. et al. 2003).....	18
Figure 4.	Seymour et al. Adjusted UTAUT Research Model From (Seymour, L. et al. 2007) .....	28
Figure 5.	Induced Model From (Khoo 2005).....	33
Figure 6.	Reasoned Innovation Model From (Swanson and Wang 2005) .....	38
Figure 7.	Components Requiring Upgrades within the 50% not meeting requirements. From (Williams 2006).....	43
Figure 8.	Percentage increases in system requirements, 2000-2006 From (Microsoft) .....	44
Figure 9.	Spending on Software Design, Development and Testing Tools From (Carmel, E. 1995).....	51
Figure 10.	Self-Reported Trade Offs for Successful Product Development From (Carmel, E. 1995).....	53
Figure 11.	Software Upgrade Cycle with Corresponding Driving Forces.....	58

THIS PAGE INTENTIONALLY LEFT BLANK

## LIST OF TABLES

Table 1.	Key data characteristics From (Venkatesh, V. et al. 2003).....	16
Table 2.	Performance expectancy: Root Constructs, Definitions, and Scales From (Venkatesh, V. et al. 2003).....	20
Table 3.	Effort Expectancy: Root Constructs, Definitions and Scales From (Venkatesh, V. et al. 2003) .....	22
Table 4.	Social Influence: Root Constructs, Definitions and Scales From (Venkatesh, V. et al. 2003) .....	24
Table 5.	Facilitating Conditions: Root Constructs, Definitions and Scales From (Venkatesh, V. et al. 2003).....	25
Table 6.	Key Factors From (Swanson and Wang 2005).....	39
Table 7.	Models A-H From (Swanson and Wang 2005) .....	41
Table 8.	Actions and Benefits for Windows Vista Deployment From (Williams 2006).....	45
Table 9.	PDMA survey on shrinking cycles in product development From (Sims, D. 1997) .....	46
Table 10.	Advantages and Disadvantages of the User Adaptation Cycle .....	59
Table 11.	Advantages and Disadvantages within the Organization Adoption Cycle .....	60
Table 12.	Advantages and Disadvantages within the Software Industry's Introduction Cycle.....	61
Table 13.	Software Industry's Cycle and Proposed Considerations and Actions.....	64
Table 14.	Organization Cycle and Proposed Considerations and Actions .....	65
Table 15.	User Cycle and Proposed Considerations and Actions .....	66

THIS PAGE INTENTIONALLY LEFT BLANK

## **ACKNOWLEDGMENTS**

We are extremely fortunate and thankful to have access to all the resources and professional expertise here at the Naval Postgraduate School. One person who stands out from the crowd, however, is Glenn Cook. We have had the privilege of being his students for several challenging courses throughout our time at NPS and we've come to learn, experience and grow from his knowledge and expertise. During one of Glenn's class lectures, he introduced the topic of this thesis and its related but disjointed cycles were discussed. He made us keenly aware that this phenomenon needed particular attention and research because thus far, the cycles had not been adequately studied in unison. He was right, we found many studies focused on user adaptation, organizational adoption or industry cycles as if they were separate and not related in significant ways. Our interest grew from what started as a 15-minute class discussion into the work and research culminating in this thesis.

We thank you Glenn, for sparking our interest, guiding us in our research and challenging us to take our thoughts into areas understudied and unknown.

THIS PAGE INTENTIONALLY LEFT BLANK

# **I. INTRODUCTION**

## **A. BACKGROUND**

Fifteen years ago most people didn't use software in their daily lives. Affordable computing didn't exist, and for those few people who did use computers as part of their daily routine, did so from expensive terminals operating proprietary software and hardware configurations, which usually required extensive training and ongoing support (Platt 2007).

That has changed completely. Almost overnight, in societal terms, cheaper hardware, software and networked computers have paved a path for the information superhighway. People use computers to pay creditors on-line using high speed internet access, send pictures, post web sites, stream video, chat, and a myriad of other processes. The modern business uses computers and software as a tool for gaining an edge over their competitors by bringing products and services to their customers in more convenient and timely ways. An explosion of information, data, resources and trade has been made possible through the globally interconnected world we know today.

Many people today find themselves using technology whether they want to or not. Computers and software present endless opportunities for the wide variety of users including the entrepreneur, CEO's, artists and young gamers everywhere. On the other hand, some users find themselves getting bogged down in the never-ending barrage of new technology implementations in their workplace. For example, many of these people might realize that their organization's new software may provide an advantage over their competitor, or may simply provide a new aspect of convenience for their customers. All too often, however, changing to new software may be anything but advantageous or convenient for those involved in the software implementation and adoption. In fact, as technology and affordability of computer systems increase and become more popular for people and their organizations, the associated software choices

too become more plentiful and complex. People trying to keep up with new advances in software might be a little overwhelmed when faced with new and improved user interfaces because it takes time to learn and adapt to the new way of doing business. In the workplace, people are expected to use many different kinds of software to carry out business as usual. In large organizations hundreds of software applications and packages are used to manage products, services and human resources and as time passes, new iterations and upgrades to these applications become available. Software manufacturers push their new software products based on speed, capabilities, ease of use and any number of other qualities that support their claims of a better version than the last one, and in many cases their claims are true. After all, this is the fundamental expectation of what we call “upgrades”.

We’ve come to understand that new software upgrades are supposed to make life easier in the workplace. A new upgrade might enable us to get work done safer, faster, and easier or in some way provide better aesthetics, which is perceived to be more professional. In any case, what we want to believe is that upgrading software will improve the business process, increase competitive advantage and provide better customer service. Unfortunately, it’s not as simple as just purchasing, installing an upgrade and receiving immediate improvements.

“We live our lives in a sea of software, but most users have no idea how software is developed or why it works the way it does. We only know that we don’t like it very much.” (Platt 2007). In addition, most users don’t know how to use software to its full potential. In many cases, software manufacturers release their software products for sale knowing significant flaws and bugs exist but in many cases, tight deadlines, budgets constraints or confidence their new software will sell anyway, entices premature releases. The release of buggy software only creates more frustration and delays for the end user contributing to resistance of future software changes in the workplace. By and large, users tend to develop their own cycle when implementing new software, but moderating factors often include training, ones resistance to the change, and the amount of time it takes him or her to become comfortable and productive with the software.

Adopting new software within organizations has cycles as well, although for different reasons than end user cycles. An organization's motives for changing, or not changing, to new software vary from such things as necessity, opportunity and affordability. For instance, an organization may upgrade to new software because not doing so would interfere with its user's ability to accomplish important business functions. Additionally, an organization might adopt new software to capitalize on an opportunity, which may lead to an advantage over its competitors. Sometimes organizations simply change to new software because they have the money to spend on what they perceive as a new and improved version. No matter the reason, organizations have cycles in adopting new software, and those cycles may be quite different from the user's cycle.

Software manufacturers also have cycles and producing and selling new software as quickly as possible for the sake of profit seems to be the name of the game in many cases. In fact, new technologies are introduced to the market at a pace that makes it difficult for organizations and their users to keep up with. The software industry's strategies rely heavily on reducing their product's time to market. Often times, quality and proper documentation take a back seat to product diversity and approaching release deadlines (Carmel, E. 1995).

Upgrades in software are inevitable and with more and more businesses adapting to computing technology in daily operations, new software has become an essential part of an organization's IT portfolio. According to a report by IDC, America's applications software market was \$54.8 billion in 2005 and is projected to increase to \$76.5 billion by 2010 (IDC 2006). Today, many organizations upgrade by purchasing and implementing packaged software. This is commercial software that is available for sale off the shelf, like MS Windows operating systems, office suites and so on. There are many kinds of software packages, ranging from end user applications to database management systems to telecommunication protocols. Additionally, outsourcing IT is becoming more and more popular as well as implementing Enterprise Resource Planning (ERP) solutions. Outsourcing and ERP solution are aimed at hiring outside organization to manage IT lifecycles, support, and upgrades.

As software companies respond to business demands and add new features to make software perform better, new versions of software are being released into the market in frequent succession (Paine 2000). In fact, between the years 1995 and 2001, Microsoft launched 6 Desktop operating systems as shown in Figure 1.



Figure 1. Windows Software Cycle  
From Windows History

<http://www.microsoft.com/windows/WinHistoryProGraphic.msp> Last accessed May 2008.

To managers everywhere, it is a constant question of whether the current version of software is “obsolete,” or outdated, and warrants an upgrade. The decision to upgrade is often times not in the hands of the end user or the organization.

Unless they are licensed to have total code autonomy, an upgrade can be just a matter of timing (Paine 2000). The question then becomes: when should they upgrade?

So how do managers within organizations determine if it's time to upgrade? If managers felt for example, that Windows XP is still sufficient for their current needs, should they migrate to the latest version – Windows Vista? What are the factors that influence their decision? What implications does it have for their organization in terms of business operating procedures, policies and doctrine? Will the manufacturer still support the older version of software that the business is running now? These are some of the questions that organizations ask every time a new version of software package emerges from the market. Deciding how and when to upgrade are not the only problems facing IS management, however, upgrades usually carry unexpected consequences as will be addressed later in the organization adoption cycle.

## **1. Known Upgrade Problems**

People often associate upgrades with better quality. Organizations usually upgrade to the latest version of software to attain efficiency and improved functionality that the current software lacks (Paine 2000). Most people would assume that simply updating or upgrading their software means they would reap the benefits of more useful features and functions, which presumably increases users' productivity. Ideally, a software upgrade will fix existing bugs and enhance operability. Unfortunately, many software upgrades have many problems upon first execution because they rarely work properly or as initially intended in the beginning. Sometimes this is due to legacy software incompatibility buggy software, or lack of knowledge.

A 2002 study commissioned by the National Institute of Standards and Technology found software bugs cost the U.S. economy about \$59.5 billion annually. The same study found that improving testing could have mitigated more than a third of that cost –about \$22.2 billion.

Although upgrades don't cause headaches for everyone everyday, routine upgrades often cause unexpected problems. Currently many people and organizations are upgrading to the Windows Vista (WV), an operating system created by Microsoft as an upgrade to Windows XP. Windows Vista was released in several stages; on November 30, 2006 it was released to business customers, computer hardware and software manufactures, and to the rest of the world on January 30, 2007. What people are finding is that Vista's Aero interface requires a specific set of video capability including: a DirectX 9 (or better) 3D graphics processor that supports 32 bits per pixel, and Pixel Shader 2.0. It must also be offered with a WDDM (Windows Vista Display Driver Model) driver. Additionally, the following hardware recommendations are common for a Vista upgrade: Intel (or comparable AMD) 1.8GHz (minimum) Pentium 4 CPU, 2GB of RAM, 80GB hard drive (60GB if you're clean installing) and a DVD drive. What this means for some people and organizations with outdated equipment is that a simple upgrade either won't work properly on their existing machines or they are faced with upgrading their hardware along with the software. Many think an upgrade is just a simple task of installing the next version of software, overlaying new code over old code, upgrading to new software can lead to many problems (Paine 2000).

As mentioned on the previous page, one of the problems inherent in packaged software is that known bugs haven't been ironed out before release. It has been found that one out of seven software firms deliver code without adequate prior testing (Minasi and Garde 1999). Another cause for upgrade problems is vendors', and their modifications of previous design logic. This often times renders the new version incompatible with the old. For example, the file format has been changed in the new Microsoft Office 2007 suite from previous versions. For older versions of Office to read the new 2007 file format, you must install the Microsoft Office Compatibility Pack for 2007 Office Word, Excel and PowerPoint file formats.

Frequent upgrades can be frustrating and troublesome to programmers, users and administrators when the upgrade produces or results in down-time, or

complications. For administrators, many man-hours are spent on correcting the faulty code or other quality user interface problems that come with an upgrade. Most organizations have spent at least some portion of their IT budget on test bed equipment, which simulates their live environment in some way. From exposing upgrades, patches and service packs to the test bed, IT professionals can catch bugs before they launch the new products in the real environment. Organizations that choose not to spend resources on testing, or can't afford to spend limited resources on testing facilities and equipment, assume great risk. Sometimes this risk is at the expense of their users and the users' productivity levels. For the unlucky, the upgrade version that was supposed to bring improvements inevitably slows down productivity when the software doesn't work properly after installation.

These problems in upgrading software can have a profound effect on the cycles associated with the new changes people and organizations are willing to make. Software is expensive especially where large organizations are concerned. Managers are therefore reluctant to purchase software if it's going to cause them problems from the start, if it's not compatible with older versions of the same software, or if they need to purchase new hardware to accommodate upgrades. For example, the British Educational Communications and Technology Agency (BECTA), responsible for advising British schools and colleges on their IT use, conducted a report on Microsoft Vista and Office 2007. Their results concluded that British schools should not make the upgrade (Becta 2008). The BECTA report recommended that British schools should not introduce the software piecemeal alongside other versions of Windows, or upgrade older machines.

"We have not had sight of any evidence to support the argument that the costs of upgrading to Vista in educational establishment would be offset by appropriate benefit," it said.

About Office 2007 it remarked, "There remains no compelling case for deployment."

BECTA warned schools that do choose to upgrade to Office 2007 should avoid Microsoft's OOXML (Office Open XML) document format because of concerns about compatibility between different applications, even though interoperability is one of the benefits Microsoft claims for the format.

We assume that reports like BECTA's affect people's motivation and organizational time intervals associated with software upgrade cycles and we intend to shed some light on why we think this is true in the next few chapters.

## **2. Importance of Upgrade Issues**

Now, we will show why this area is worthy of further study. First, it is unknown how carefully organizations are paying attention to software upgrade problems. Until the problems are identified and studied it seems that no viable action will emerge toward solving the issues. Second, software upgrades are a continuous problem because once software is installed; users and organizations will very likely want an upgrade eventually unless it is completely abandoned. Third, the actual impacts of software upgrades are largely unknown. Anecdotal evidence suggests that the upgrade process is problematic, however, to date, we have been unsuccessful in finding evidence that attempts to fully understand the problems. Fourth, it is unknown how organizational cycles of upgrades are coping with the problems or preparing them for the next upgrade. Fifth, If the phenomenon of software upgrade cycles can be understood, then better strategies can be developed which might help users adapt easier, organizations adopt more effectively and software manufacturers to develop and deploy more efficiently.

Overall, this is an area that in our opinion has largely escaped academic attention. Most investigations of the problems described above are gathered from trade journals while no empirical research has been found that researches these issues. Traditionally, academic research has focused primarily on studies related to information systems implementation (Lucas, Walton et al. 1988; Thong,

Yap et al. 1996) and no studies were found to have concentrated solely on software upgrade cycles among people, organizations and the software industry.

One possible reason for this lack of academic attention could be that it has been overlooked as a trivial problem. If the software already exists and an upgrade is merely providing subtle changes, then user acceptance or technology fit may not be as big of an issue. These kinds of upgrades may not incur as much business process change or adaptation time as completely new software suite integration would. Still, as the problems presented in the previous chapters indicate, unanticipated problems do occur when new software is installed.

Implementing new software upgrades warrant further research. The objective of this study is to investigate the cyclical phenomenon associated with new software upgrades from three standpoints: people's ability to adapt to new software, organizations ability to adopt and install new software, and the software industry's desires to introduce new software. These aspects will be compared to understand and identify possible benefits and value among re-occurring cycles in software implementation within organizations.

The following research questions are posed to investigate these phenomena.

1. Primary: How are the organizational, personal and software industry's development and adoption cycles related?
2. Secondary 1: How might synchronization of cycles benefit adoption of software?
3. Secondary 2: Is technology, with respect to advances in software, moving at a pace that is too fast for large organizations like the DoD to keep up with?

THIS PAGE INTENTIONALLY LEFT BLANK

## **II. METHODOLOGY**

### **A. QUALITATIVE APPROACH**

To investigate the research questions, a qualitative approach was deemed most appropriate. First, unlike some phenomena that can be simulated and studied in a laboratory environment, software upgrades and how they affect users, organizations and the software industry cannot be studied in a similar environment. The phenomenon our research is based upon can be studied in organizations where software upgrades were implemented, where people adapted to upgrades and where the software industry was able to introduce upgrades. According to Denzin and Lincoln (1998),

“Qualitative research is multi-method in focus, involving an interpretive, naturalistic approach to its subject matter. This means that qualitative researchers study things in their natural settings, attempting to make sense of, or interpret, phenomena in terms of the meanings people bring to them.”

Qualitative research also has four characteristics according to Lee, Mitchell et al. (1999) It occurs in a natural setting, it derives data from the perspective of participants, it can be flexibly changed to accommodate the demands the research situation, and last, it does not have standard instrumentation, which makes notions of control, reliability and validity difficult to obtain.

The objective of this research is to study software upgrades where they occur from three perspectives:

The organizational perspective: Where software upgrade adoption cycles are disjointed from user cycles and software production cycles.

The user perspective: Where software upgrade adaptation cycles are disjointed from their organization's cycle as well as software production cycles.

The software industry's perspective: Where production cycles are disjointed from average organizational and user cycles.

Additional areas of interest will be researched in an attempt to provide at least partial answers our secondary research questions mentioned in the last chapter. It is assumed that as technology allows software to progress and be produced in faster cycles, and that organizations and their users are less likely to be capable of making use of these rapidly evolving software benefits. Additionally, we believe that as these cycles increase in speed and complexity, those who are unable or incapable of evolving with new software trends will be more likely to lose vital competitive advantages.

## **B. RESEARCH DESIGN**

From a research standpoint, our design is simplistic which allows for a loose structure. A research design that is tight in structure is usually characteristic of one that also has many pre-determined guidelines. Consequently, our first question was to ask our selves how tight the preliminary design should be and how much planning should go into it.

Some factors are important in trying to answer this question. First, it depends on the nature of the research involved: whether the research is exploring and understudied phenomenon, to induce a social theory, to confirm well-defined constructs, or to investigate a hypothesis. Traditionally, a loose structure in research design is usually common with exploratory, inductive research whereas a tightly structured research design is usually synonymous with confirmatory, theory testing type of research. Miles and Huberman (1994) claim:

Much qualitative research lies between these two extremes. Something is known conceptually about the phenomena, but not enough to house a theory. The researcher has an idea of the parts of the phenomenon that are not well understood and knows where to look for these things – in which settings, among which actors. And the researcher usually has some initial ideas about how to gather the information. At the outset, then, we usually have at least

a rudimentary conceptual framework, a set of general research questions, some notion about sampling, and some initial data-gathering devices. (p. 17)

### C. CONCEPTUAL DESIGN

“A conceptual framework explains, either graphically or in narrative form, the main things to be studied” (Miles and Huberman 1994). The “main things” mentioned here refer to the key elements of the research being conducted and the relationships among them. In this case, our conceptual framework does not represent a theoretical model awaiting confirmation. Instead, it is a map of those elements that are to be explored in this study. The preliminary conceptual framework is presented in Figure 2 below.

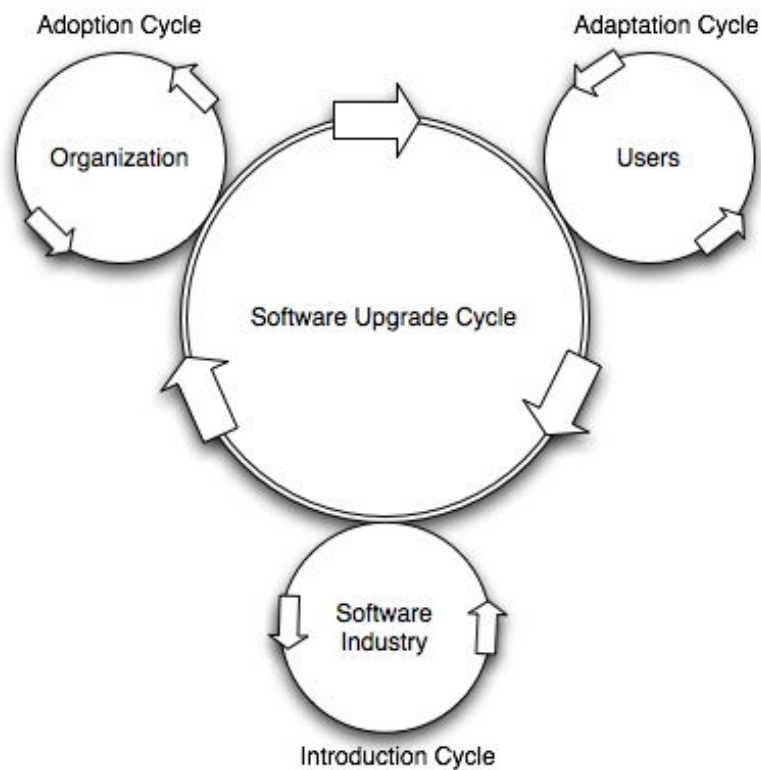


Figure 2. Software Upgrade Cycle

## 1. Description of the Diagram

There are four circles in Figure 2, Organization, Users, Software Industry and Software Upgrade Cycle. The large circle in the middle of the diagram represents the main topic (Software Upgrade Cycle) of our study and the smaller circles encompassing it represent the basis of our research in examining the main topic. Each smaller circle is labeled, on its outer edge, with the name of the cycle associated with the area it focuses on. For example, the User circle is an area of proposed study and it is labeled with Adaptation Cycle, which represents the problem or issue that we assume effects the user in the most meaningful way.

Here, our ideas are represented in the form of circles with arrows because we want to display our interpretation that software upgrade cycles are a process model rather than a causal model. The smaller circles are also touching the larger circle, which is intended to represent the idea that organizations, users and the software industry are very much linked to the software upgrade cycle process. We suspect also that the internal and external influences that drive the organizations, users and software industry, also drive the software upgrade cycle. From this idea, we can “see” the smaller circles grinding away at the larger circle and spinning it. This alludes to the fact that as the cycles of the organizations, users and software industry increase in speed, so will the entire software upgrade cycle process speed up.

As one can imagine, the circles spinning in an orderly or “gear-driven” fashion, it is important to recognize that we don’t believe this process is actually orderly. These cycles are in fact disjointed and act separately as internal and external forces affect them. Our figure was purposely drawn with circles instead of gears to highlight the fact that slippage occurs and the circles can spin at very different speeds in comparison with the other circles. This represents the idea that, for instance, the software industry cycle may be churning at a faster speed than the organization cycle. This simply indicates that the rate at which software is being introduced is outpacing the rate at which an organization is able to adopt that software.

### **III. USERS, ORGANIZATIONS AND SOFTWARE INDUSTRY**

#### **A. USER ADAPTATION CYCLES**

Recent research focusing on users' adaptation to new technology is predominantly studied using two commonly used theories, the Unified Theory of Acceptance and Use of Technology (UTAUT) and the Technology Acceptance Model (TAM). We will first describe these two theories and reveal assumptions predicated on strong behavioral intentions, usage behaviors, perceived usefulness and perceived ease of use. We will then reference the UTAUT in attempting to explain adaptation behaviors of users in organizations through exploring two case studies. The Technology Acceptance Model has been a popular method in researching and realizing the importance of end-user acceptance, a key success factor of Enterprise Resource Planning (ERP) implementations. Criticism against applying the TAM to examine ERP acceptance is that the use of an ERP is mandatory for an organizations end users, while an implicit assumption of TAM is that users of the information systems have some level of choice with regard to the extent that they use the technology. For this reason, we will focus on using case studies that use the UTAUT model instead of the TAM. Researchers have found that using TAM to evaluate acceptance of new technology provides limited explanation of end-users' behavior, attitudes and perceptions towards the system and that the results of studies based on the TAM may provide misleading recommendations for organizations (Brown et al. 2002).

##### **1. Exploring the UTAUT Model**

The first case study we will focus on is called "User Acceptance of Information Technology: Toward a Unified View" by Venkatesh, Morris, Davis, B., and Davis, F. This case study reviews and consolidates constructs and extensions from eight prominent models of usage behavior to formulate the

Unified Theory of Acceptance and Use of Technology model. The study used data from four organizations over a six-month period, among individuals being introduced to a new technology in the workplace. Samples were gathered for heterogeneity across technologies, organizations, industries, business functions and nature of use (voluntary vs. mandatory). The approach used by Venkatesh et al. in gathering and validating data was consistent with prior training and individual acceptance research where individual reactions to a new technology were studied (e.g., Davis et al. 1989; Olfman and Manviwalla 1994; Venkatesh and Davis 2000). The methodology for acquiring the data came from a pre-tested questionnaire containing items measuring constructs from the eight models at three different points in time: post-training, one month after implementation, and three months after implementation. Table 1 summarizes the key characteristics of the organizational settings.

Industry	Functional Area	Sample Size	System Description
<b>Voluntary Use</b>			
Entertainment	Product Development	54	Online meeting manager that could be used to conduct web-enabled video or audio conferences in lieu of face-to-face or traditional phone conferences
Telecomm Services	Sales	65	Database application that could be used to access industry standards for particular products in lieu of other resources (e.g., technical manuals, Web sites)
<b>Mandatory Use</b>			
Banking	Business Account Management	58	Portfolio analyzer that analysts were required to use in evaluating existing and potential accounts
Public Administration	Accounting	38	Proprietary accounting systems on a PC platform that accountants were required to use for organizational bookkeeping

Table 1. Key data characteristics  
From (Venkatesh, V. et al. 2003)

In using UTAUT and TAM, we expect to be able to shed further light on how and why users in an organization adapt to new software. The case studies will confirm that the theories provide a useful tool for managers needing to assess the likelihood of success for new technology introductions and help them understand the drivers of acceptance within their organizations. This ultimately gives managers an advantage towards proactively designing interventions (including training, marketing, etc.) targeted at populations of users that may be less inclined to adopt and use new systems.

First, it is important to understand the principles of the theories described above in order to provide a foundation in understanding users' adaptation of new software. We will begin by exploring the Unified Theory of Acceptance and Use of Technology. The UTAUT aims to explain user intentions to use an Information Systems (IS) and subsequent usage behavior. The theory holds that four key constructs (performance expectancy, effort expectancy, social influence, and facilitating conditions) are direct determinants of usage intention and behavior (Venkatesh et al., 2003). Gender, age, experience, and voluntariness of use are posited to mediate the impact of the four key constructs on usage intention and behavior (Venkatesh et. al., 2003). The theory was developed through a review and consolidation of the constructs of eight models that earlier research had employed to explain IS usage behavior (theory of reasoned action, technology acceptance model, motivational model, theory of planned behavior, a combined theory of planned behavior/technology acceptance model, model of PC utilization, innovation diffusion theory, and social cognitive theory). Subsequent validation of UTAUT in similar studies found it to account for 70% of the variance in usage intention (Venkatesh et. al., 2003).

Of the eight models listed above, Venkatesh et al., theorizes that four of the constructs play a significant role as direct determinants of user acceptance behavior. These are performance expectancy, effort expectancy, social influence and facilitating conditions. Each of the determinants is impacted by the role of their key moderators, which are gender, age, voluntariness and experience. Exploring these determinants and their key moderators reveal theoretical

justification in explaining the phenomena about how and why users in an organization adapt to new software. Figure 3 shows the Venkatesh et al. research model, which depicts the four determinants and their four key moderators. Note that the model shows these constructs as directly influencing a user's behavioral intention and use behavior.

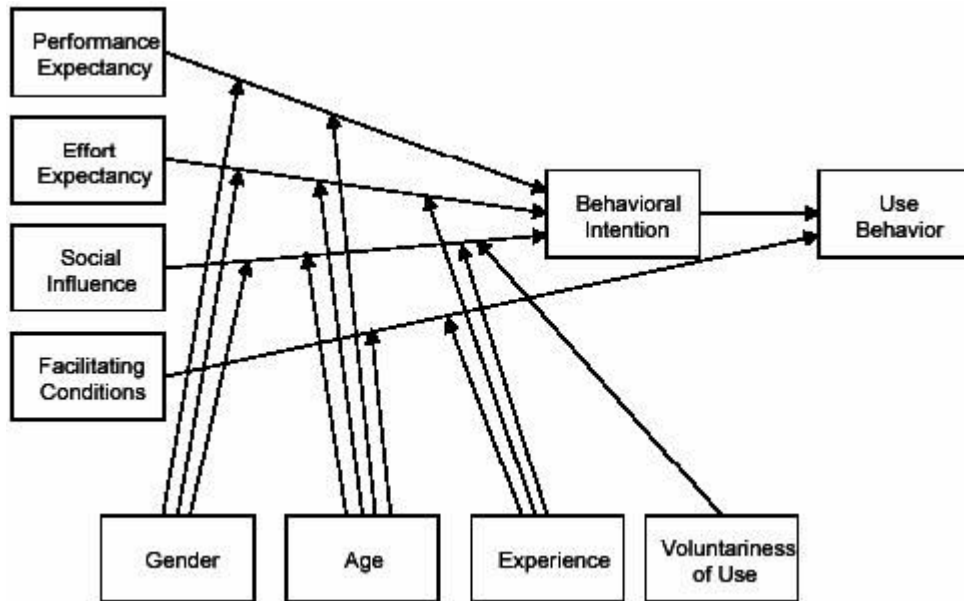


Figure 3. UTAUT Research Model  
From (Venkatesh, V. et al. 2003)

The four key constructs (performance expectancy, effort expectancy, social influence and facilitating conditions) according to Vanketesh et al., are used to explain significant direct determinants of user acceptance and usage behavior. In the remainder of this section we define each of the determinants, specify the role of the key moderators (gender, age, voluntariness, and experience), and provide the theoretical justification relating to user adaptations to new software and subsequent usage behavior.

**a.     *Performance Expectancy***

Performance expectancy is defined as the degree to which an individual believes that using the system will help him or her to attain gains in job performance. The performance expectancy construct was found to be the strongest predictor of intention and remained a significant indicator at all points of measurement in both voluntary and mandatory settings within the Venkatesh et al. case study. Five constructs pertaining to performance expectancy were consolidated from the eight previously mentioned models, they are:

- perceived usefulness
- extrinsic motivation
- job-fit
- relative advantage
- outcome expectation

Table 2 describes each of these root constructs and describes the scales used for the study. Within each scale is a list of items that correspond to perceptions on the performance expectancy realized by each individual user. These items are of particular interest as they relate to explanations regarding usage behavior and intent to use the new technology.

<b>Performance Expectancy: Root Constructs, Definitions and Scales</b>		
Perceived Usefulness	The degree to which a person believes that using a particular system would enhance his or her job performance	<ol style="list-style-type: none"> <li>1. Using the systems in my job would enable me to accomplish tasks more quickly.</li> <li>2. Using the system would improve my job performance.</li> <li>3. Using the system in my job would increase my productivity.</li> <li>4. Using the system would enhance my effectiveness on the job</li> <li>5. Using the system would make it easier to do my job</li> <li>6. I would find the system useful in my job.</li> </ol>
Extrinsic Motivation	The perception that users will want to perform an activity because it is perceived to be instrumental in achieving valued outcomes that are distinct from the activity itself, such as improved job performance, pay, or promotions	Extrinsic motivation is operationalized using the same items as perceived usefulness (items 1 through 6 above)
Job-fit	How the capabilities of a system enhance an individual's job performance	<ol style="list-style-type: none"> <li>1. Use of the system will have no effect on the performance of my job.</li> <li>2. Use of the system can decrease the time needed for my important job responsibilities</li> <li>3. Use of the system can significantly increase the quality of output on my job</li> <li>4. Use of the system can increase the effectiveness of performing job tasks</li> <li>5. Use can increase the quantity of output for the same amount of effort</li> <li>6. Considering all tasks, the general extent to which use of the system could assist on the job. (different scale used for this item)</li> </ol>
Relative advantage	The degree to which using an innovation is perceived as being better than using its precursor	<ol style="list-style-type: none"> <li>1. Using the system enables me to accomplish tasks more quickly</li> <li>2. Using the system improves the quality of the work I do.</li> <li>3. Using the system makes it easier to do my job</li> <li>4. Using the system enhances my effectiveness on the job</li> <li>5. Using the system increases my productivity</li> </ol>
Outcome expectations	Outcome expectations relate to the consequences of the behavior	<p>If I use the system...</p> <ol style="list-style-type: none"> <li>1. I will increase my effectiveness on the job</li> <li>2. I will spend less time on routine job tasks</li> <li>3. I will increase the quality of output of my job</li> <li>4. I will increase the quantity of output for the same amount of work</li> <li>5. My coworkers will perceive me as competent</li> <li>6. I will increase my chances of obtaining a promotion</li> <li>7. I will increase my chances of getting a raise</li> </ol>

Table 2. Performance expectancy: Root Constructs, Definitions, and Scales  
From (Venkatesh, V. et al. 2003)

Venkatesh et al., expected that from a theoretical standpoint, the relationship between performance expectancy and intention are moderated by gender and age. Research on gender differences have indicated that men tend to be more task oriented than women (Minton and Schneider 1980) and, therefore, performance expectancies, which mainly focus on task accomplishment, are likely to be especially prominent to men. Similar to gender,

age is theorized to play a moderating role (Venkatesh et al. 2003). Previous research on job related attitudes suggest that younger workers may place more importance on extrinsic rewards (Hall and Mansfield 1995). It is also important to note that studies of gender differences can be misleading without reference to age. A good example points towards the traditional societal gender roles where job related factors may change significantly (e.g., become supplanted by family-oriented responsibilities) for working women between the time that they enter the labor force and the time they reach child-rearing years. Thus, we surmise that gender and age both influence performance expectancy such that the effect tends to be stronger for men and particularly for younger women.

***b. Effort Expectancy***

Effort expectancy is defined as the degree of ease associated with the use of the system. The effort expectancy construct is significant in both voluntary and mandatory usage contexts but tends to become less significant over periods of extended and sustained usage. Venkatesh et al. concludes that the influence of effort expectancy on behavioral intention is moderated by gender, age and experience, such that the effect is stronger for women, particularly younger women, and particularly at early stages of experience. Table 3 describes each of these root constructs and describes the scales used for the study. Within each scale is a list of items that correspond to perceptions on the effort expectancy realized by each individual user. These items are of particular interest as they relate to explanations regarding usage behavior and intent to use the new technology.

<b>Effort Expectancy: Root Constructs, Definitions and Scales</b>		
Percieved Ease of Use	The degree to which a person believes that using a system would be free of effort	<ol style="list-style-type: none"> <li>1. Learning to operate the system would be easy for me.</li> <li>2. I would find it easy to get the system to do what I want it to do.</li> <li>3. My interaction with the system would be clear and understandable</li> <li>4. I would find the system to be flexible to interact with</li> <li>5. It would be easy for me to become skillful at using the system</li> <li>6. I would find the system easy to use</li> </ol>
Complexity	The degree to which a system is perceived as relatively difficult to understand and use	<ol style="list-style-type: none"> <li>1. Using the system takes too much time from my normal duties.</li> <li>2. Working with the system is so complicated, it is difficult to understand what is going on</li> <li>3. Using the system involves too much time doing mechanical operations (e.g., data input)</li> <li>4. It takes too long to learn how to use the system to make it worth the effort</li> </ol>
Ease of Use	The degree to which using an innovation is perceived as being difficult to use	<ol style="list-style-type: none"> <li>1. My interaction with the system is clear and understandable</li> <li>2. I believe that it is easy to get the system to do what I want it to do.</li> <li>3. Overall, I believe that the system is easy to use.</li> <li>4. Learning to operate the system is easy for me.</li> </ol>

Table 3. Effort Expectancy: Root Constructs, Definitions and Scales  
From (Venkatesh, V. et al. 2003)

Effort expectancy has been found to be more salient for women than for men (Venkatesh and Mooris 2000) and (Bem and Allen 1974). Similar to performance expectancy, gender differences are commonly driven by gender roles. Increased age has been found to be associated with difficulty in processing complex stimuli and allocating attention to information on the job, both of which may be necessary when using software systems (Plude and Hoyer 1985).

### **c. Social Influence**

Social influence is defined as the degree to which an individual perceives that important others believe he or she should use the new system. Each construct associated with social influence contains the explicit or implicit notion that the individual's behavior is influenced by the way in which they believe others will view them as a result of having used the technology. Social influence in technology acceptance decisions is complex and subject to a wide range of contingent influences. Intention to use and behavior studies show that

none of the social influence constructs are significant in voluntary contexts; however, each becomes significant when use of a system is mandatory. Venkatesh et al. concludes that in mandatory settings, social influence appears to be important only in the early stages of individual experience with the technology, but its role erodes over time and eventually becomes nonsignificant with sustained usage. Social influence has an impact on individual behavior through three mechanisms: compliance, internalization, and identification. The compliance mechanism causes an individual to simply alter his or her intention in response to social pressure (i.e., the individual intends to comply with the social influence). Internalization and identification relate to altering an individual's belief structure and/or cause an individual to respond to potential social status gains. Prior research suggests that individuals are more likely to comply with others' expectations when those referent others have the ability to reward the desired behavior or punish nonbehavior (Warshaw 1980). Technology acceptance studies indicate that reliance on others' opinions is also significant only in mandatory settings and particularly in the early stages of experience, when an individual's opinion may be relatively ill-informed or premature (Venkatesh and Davis 2000). Venkatesh et al., theorizes that women tend to be more sensitive to others' opinions and therefore find social influence to be more salient when forming an intention to use new technology. Additionally, older workers have been found to be more likely to place increased salience on social influences, with the effect declining as more experience is gained. Thus, the influence of social influence on behavioral intention is moderated by gender, age, voluntariness and experience, such that the effect will be stronger in women, particularly older women, and particularly in mandatory settings in the earlier stages of experience.

Table 4 describes each of the root constructs associated with social influence and describes the scales used for the study. Within each scale is a list of items that correspond to perceptions on the effort expectancy realized by each individual user. These items are of particular interest as they relate to explanations regarding usage behavior and intent to use the new technology.

<b>Social Influence: Root Constructs, Definitions and Scales</b>		
Subjective Norm	The person's perception that most people who are important to him think he/she should or should not perform the behavior in question	<ol style="list-style-type: none"> <li>1. People who influence my behavior think that I should use the system</li> <li>2. People who are important to me think that I should use the system</li> </ol>
Social Factors	The individual's internalization of the reference group's subjective culture, and specific interpersonal agreements that the individual has made with others, in specific social situations	<ol style="list-style-type: none"> <li>1. I use the system because of the proportion of coworkers who use the system</li> <li>2. The senior management of this business has been helpful in the use of the system</li> <li>3. My supervisor is very supportive of the use of the system for my job</li> <li>4. In general, the organization has supported the use of the system</li> </ol>
Image	The degree to which use of an innovation is perceived to enhance one's image or status in one's social system	<ol style="list-style-type: none"> <li>1. People in my organization who use the system have more prestige than those who do not</li> <li>2. People in my organization who use the system have a high profile</li> <li>3. Having the system is a status symbol in my organization</li> </ol>

Table 4. Social Influence: Root Constructs, Definitions and Scales  
From (Venkatesh, V. et al. 2003)

#### ***d. Facilitating Conditions***

Facilitating conditions are defined as the degree to which an individual believes that an organizational and technical infrastructure exists to support the use of a system. The concept is characterized by three different constructs: perceived behavioral control, facilitating conditions and compatibility. Each of these concepts includes aspects of the technological and/or organizational environment that are designed to remove barriers to use. Table 5 describes each of the root constructs associated with facilitating conditions and describes the scales used for the study. Within each scale is a list of items that correspond to perceptions on the effort expectancy realized by each individual user. These items are of particular interest as they relate to explanations regarding usage behavior and intent to use the new technology.

<b>Facilitating Conditions: Root Constructs, Definitions and Scales</b>		
Perceived Behavioral Control	Reflects perceptions of internal and external constraints on behavior and encompasses self-efficacy, resource facilitating conditions, and technology facilitating conditions	<ol style="list-style-type: none"> <li>1. I have control over using the system</li> <li>2. I have the resources necessary to use the system</li> <li>3. I have the knowledge necessary to use the system</li> <li>4. Given the resources, opportunities and knowledge it takes to use the system, it would be easy for me to use the system</li> <li>5. The system is not compatible with other systems I use</li> </ol>
Facilitating Conditions	Objective factors in the environment that observers agree make an act easy to do, including the provision of computer support	<ol style="list-style-type: none"> <li>1. Guidance was available to me in the selection of the system</li> <li>2. Specialized instruction concerning the system was available to me</li> <li>3. A specific person (or group) is available for assistance with the system difficulties</li> </ol>
Compatibility	The degree to which an innovation is perceived as being consistent with existing values, needs, and experiences of potential adopters	<ol style="list-style-type: none"> <li>1. Using the system is compatible with all aspects of my work</li> <li>2. I think that using the system fits well with the way I like to work</li> <li>3. Using the system fits into my work style</li> </ol>

Table 5. Facilitating Conditions: Root Constructs, Definitions and Scales  
From (Venkatesh, V. et al. 2003)

Perceived behavioral control is significant in both voluntary and mandatory settings especially immediately after training is conducted on a new system but tends to become non-significant by at least 1 month after implementation. Facilitating conditions have been found not to have a significant influence on behavioral intention but do seem to have a direct influence on usage beyond factors explained by behavioral intentions alone. Facilitating conditions are also modeled as a direct antecedent of usage (i.e. not fully mediated by intention). In fact, the effect is generally expected to increase with experience as users of the new technology find different avenues for help and support throughout the organization. Studies in organizational psychology have found that older workers attach more importance to receiving help and assistance on the job (Hall and Mansfield 1995). Thus, Venkatesh et al. concludes, when moderated by experience and age, facilitating conditions have a significant influence on usage behavior. This effect becomes stronger for older workers, particularly with increasing experience.

From a theoretical perspective, UTAUT provides a refined view of how the determinants and usage behavior evolve over time. It is important to note that most of the key relationships of the model are moderated. Age has

received very little attention in the study of technology acceptance research literature, yet numerous recent studies like UTAUT show that age moderates all of the key relationships. Gender, which has received more recent attention, is also a key moderating influence, however, it appears to work in concert with age. As more studies in sociology and psychology appear in an IT context, it is apparent that as the younger population of our workforce matures, gender differences, in how each perceives information technology may disappear. This is a hopeful sign and suggests that gender differences in the use of information technology may be transitory, especially concerning the younger generation of workers raised and educated in today's digital age.

## **2. Exploring a Revised UTAUT Model**

As discussed earlier in this section, we intend to use a second case study wherein UTAUT is used to explain users' acceptance of new technology. In End-Users' Acceptance of Enterprise Resource Planning Systems: An Investigation of Antecedents by Symour, L., Makanya, W., and Berrange, S., this phenomenon is further studied based on user acceptance of an ERP implementation at the University of Cape Town, South Africa. Symour et al., propose a research model that is an adjustment to the UTAUT and contains the dependant variable Symbolic Adoption that has been shown to better indicate end-user acceptance of mandatory technologies by Nah, Tan and Teh (2004). The research model was validated using a 2006 survey of users of the PeopleSoft Student Administration System at the University of Cape Town. The PeopleSoft system is mainly used for the maintenance of access to student related data and for student processes such as registration and graduation.

Before we discuss the research methodology, data analysis and results of the study, it is important to briefly introduce the concept of implementing an ERP. Enterprise Resource Planning systems are made up of a suite of integrated software applications designed to support a business' core functions. ERP systems help organizations reduce operating expenses and are intended to improve business process management through the integration of business

functions and information. Despite the advantages an ERP solution may provide, research finds that their adoption is often problematic. Approximately 50% of all ERP implementations fail to meet the adopting organizations' expectations (Jasperson, Carter and Zmud 2005). End user resistance is one of the main contributing factors that lead to the failure of an ERP adoption and it has been found that user acceptance of a system is highly dependant on attitude towards the information system, their overall acceptance of the system, and the level of intended usage of the system. Examining these factors is therefore important for organizations that intend to or are installing ERP systems.

Like the study conducted by Venkatesh et al., the Seymour et al. study used the UTAUT model with a few adjustments. The behavioral intention factor and the use behavior factor used in UTAUT were replaced with the symbolic adoption factor since it was viewed to provide a more suitable measure of end-users' acceptance from an ERP perspective. To date the UTAUT model, in a strict sense, has not been used in validating an ERP implementation. Symbolic adoption, first hypothesized by Rawstorne et al. (1998), is used as an improved dependant variable and correlation between symbolic adoption and perceived ease of use. Symbolic adoption can be described as an end-user's "mental acceptance of a new system (Nah et al. 2004). It is further suggested that end-users in a mandatory setting undergo symbolic adoption before actual system acceptance takes place. In a mandatory setting, end-users will demonstrate differences in symbolic adoption, which can be further investigated to evaluate and explain their acceptance levels of ERP systems. Figure 4 below represents the research model used by Symour et al. and can be compared to the Venkatesh et al. UTAUT model. Note that training, project communication and shared belief are used as facilitating conditions effecting symbolic adoption, where in the UTAUT model, facilitating conditions effect use behavior.

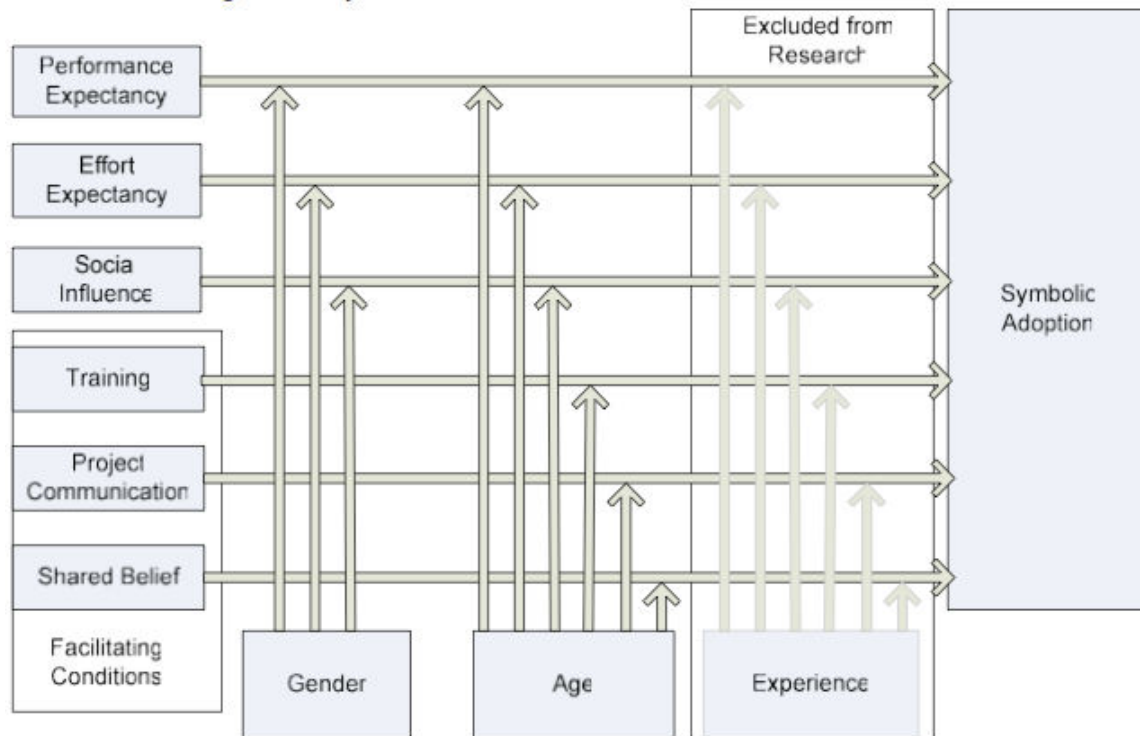


Figure 4. Seymour et al. Adjusted UTAUT Research Model  
From (Seymour, L. et al. 2007)

Training, project communication and shared belief are now briefly described below to shed further light on the Seymour et al. facilitating conditions methodology.

#### **a. Training**

Training has been identified as an important factor for implementing ERP systems and focuses generally on training length, timing and thoroughness. It is important to train users on a new ERP system because of the proven positive influence on end-user acceptance of the system. Training gives end-users time to adjust to the change that will occur and allows them to gain first hand experience and explore the usefulness of the system. Brown et al. (2002) noted how training increases the self-efficacy of end-users of ERP systems because they understand better how the system may improve their job functions.

**b. Shared Belief**

Shared belief is defined as the end-users' belief that the ERP system will have an overall positive effect on the organization. End users belonging to different functional areas of an organization make use of ERP systems when they believe those systems are going to help them integrate the organizations different business functions. The caveat is that all end-users of the ERP system must therefore believe that the ERP system will benefit the organization before it is accepted. Prior research in end-user acceptance of an ERP system indicates that if all end-users have a shared belief and an understanding of why the system is being implemented, including how it would benefit the organization and improve their work environment, the system will be more readily accepted.

**c. Communication**

An important factor in increasing user acceptance of an ERP implementation is early and effective communication with end-users. It is important for organizations to clearly justify the system's benefits to establish a shared belief among the end-users that the system will provide better functionality within the organization. Doing this has been shown to curb end-user resistance and facilitate acceptance of an ERP (Oliver, D., Whymark, G., and Rohm, C. 2005). End-users who feel they are included in the decision to adopt an ERP system from the beginning, are more likely to accept it because communication provides information about the system and allows users to provide feedback on issues they may have. Thus, resistance to the new system is detected early and measures can be taken to counteract and correct aspects, which may negatively impact the implementation effort.

Results from the Seymour et al. study correlate with the Venkatesh et al. study in that the UTAUT model could validate relationships among end user acceptance of a new system. In the Seymour et al. study, the independent variables accounted for 79% of the total variation in the dependant variable,

symbolic adoption, for older respondents. This percentage went down to 56% for the total group and further down to 39% for the younger respondents (where age >35 years old). In support of the literature, positive linear correlation was found between performance expectancy; effort expectancy; project communication; training and shared belief, and the dependant variable symbolic adoption. Together, as stated above, the models relationships accounted for 56% of the variation in symbolic adoption. While this is a good result, it also indicates that there are other variables not tested within this study which impact symbolic adoption. This study shows that similar supporting relationships exist between slightly different UTAUT models. Below is a list of the correlating relationships found in the Seymour et al. study. Notice the similarities of the relationships described in the Venkatesh et al. study.

1. The relationship between effort expectancy and symbolic adoption is influenced by age, such that the effect will be stronger for younger respondents.
2. The relationship between performance expectancy and symbolic adoption is influenced by age, such that the effect will be stronger for younger respondents.

It was determined that age is a significant moderating factor between effort expectancy and usage of the system but that age was not a significant moderating factor between performance expectancy and system use. Support was also found for the following three moderating relationships:

1. The relationship between shared belief and symbolic adoption is influenced by age, such that the effect will be greater for older workers.
2. The relationship between project communication and symbolic adoption is influenced by age, such that the effect will be greater for older workers
3. The relationship between training and symbolic adoption is influenced by age, such that the effect will be greater for older workers.

The findings of both the Venkatesh et al. study and the Seymour et al. study have high correlating relationships with respect to age. Gender was unable to be used as a moderating variable in the Seymour et al. study because there were only 3 male respondents in the data sample. Additionally the Seymour et al. study was not conducted as a longitudinal study and therefore levels of acceptance could not be validated at different points in the implementation. These two points limited the scope of the study, however, it is abundantly clear that the UTAUT model does validate the use of the four key constructs (Performance Expectancy, Effort Expectancy, Social Influence and Facilitating Conditions) and their moderators (Gender, Age, Experience and Voluntariness of Use) despite slight variance in use of the model in the case of Semour et al.

Both studies point to the fact that further research in this area is critical in developing a better understanding on user acceptance of new technology. The findings in both studies indicate the relevance to future researchers and leaders within organizations intending to implement new systems. It is arguable among practitioners today, that getting end-users to use a new system correctly may be much harder and more important than actually implementing the system itself because the success of a system is generally measured on how well the system is used by end-users.

## **B. ORGANIZATION ADOPTION CYCLES**

According to Moore's Law, the number of transistors on a chip will double every two years (Moore 1965). Moore's law can really be thought of as a forecast, revealing the continuous evolution, change and improvements in information systems technology. Constant changes can be overwhelming within a business organization attempting to maintain a competitive edge in an ever-expanding marketplace. An organization's motives for adopting or not adopting new technologies vary from such things as necessity, opportunity and affordability. Worth mentioning is the fact that there are great many other motivating factors which moderate an organizations motives to adopt new

technology. For instance, an organization may upgrade to new software because not doing so would interfere with its user's ability to accomplish key business processes. Additionally, an organization might adopt new software to capitalize on an opportunity that may lead to an advantage over its competitors. Sometimes organizations simply change to new software because of growing IT budgets coupled with perceived benefits in new and improved product versions. Another significant factor that entices organizations to upgrade old software is software manufacturer support and the licensing of their software products. Regardless of the reasons for change, upgrades in technology and in particular, software, are to be expected.

Furthermore, in order for a business to survive in today's fast paced corporate environment, their value chain will inevitably incorporate some sort of computing technology in daily operations. In fact, technology is embedded in every value activity in a firm, and technological change can affect competition through its impact on virtually any activity (Porter, M. 1985). These aspects have been thoroughly studied and documented and are in part, a motivating factor behind new software adoption practices essential in every organization.

With the understanding that software will continue to change and improve, business managers struggle with the question of whether the current version of software is obsolete or outdated, and warrants an upgrade. The rest of this section will outline some of the critical questions that a manager must ask to determine if an organization needs to adopt new software, followed by some possible coping strategies when the choice has been made to do so.

## **1. Critical Software Upgrade Questions**

How do managers within organizations determine if it's time to upgrade? If managers felt that Windows XP for example, was still sufficient to satisfy their current business needs, should the organization still consider changing to the latest version of Windows Vista anyway? What are the factors, which might influence their decision? What implications does it have for their organization in

terms of the realignment of business operating procedures, processes, policies and doctrine? How much longer will the manufacturer continue to support the older version of software that the business is currently running and how does this timeframe fit within the organizations schedule and operations? These are some of the questions that organizations ask every time a new version of software package emerges from the market. But, deciding if and when to upgrade are not the only problems facing Information Systems (IS) management. They also have to be concerned with unexpected consequences that may occur due to the upgrade. Figure 5 below helps to identify key factors that can influence a software upgrade.

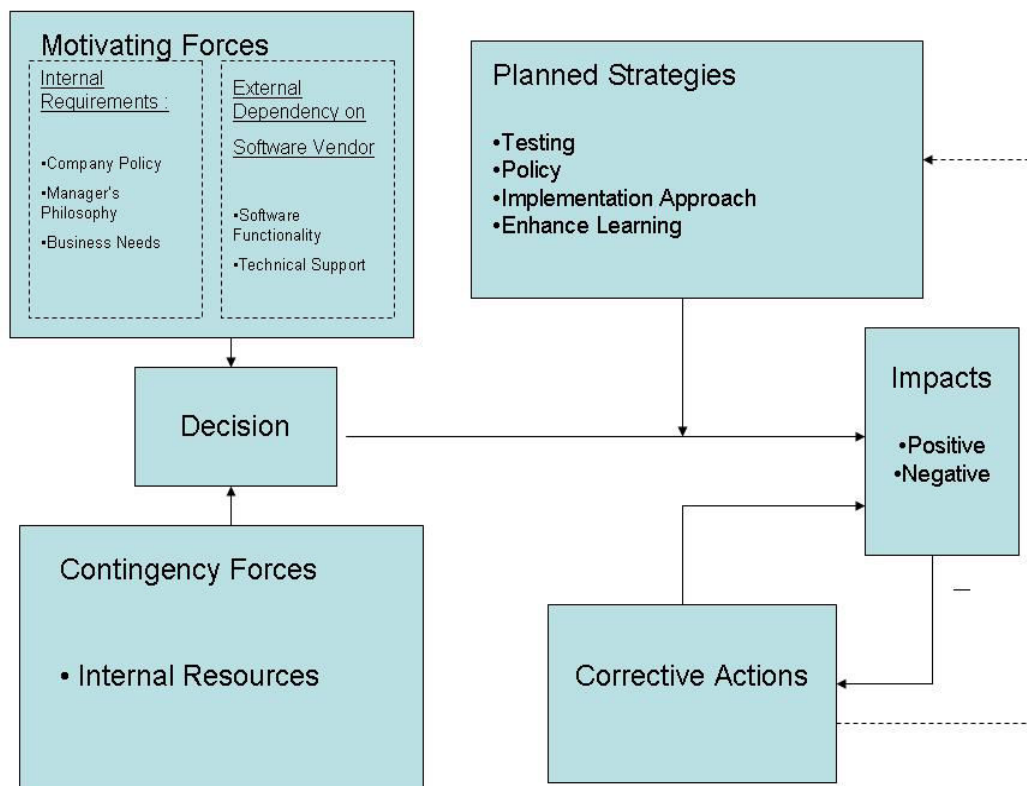


Figure 5. Induced Model  
From (Khoo 2005)

According to Khoo, M., Huoi the six components in this model are; Motivating Forces, Decisions, Contingency Forces, Planned Strategies, Impacts and Corrective Actions. These components help to identify some of the key factors that drive software upgrades within an organization.

**a. *Motivating Forces***

Motivating Forces can be broken down into two main categories: Internal Requirements, and External Dependency on Software Vendors. The Internal Requirements consist of:

- Company Policy

Organizations should develop a policy on software upgrades that helps to provide a step-by-step process to guide owners, managers, and stakeholders on what when and how to upgrade. It should govern what type of software should be adapted, when it is necessary, and guide the IT manager on choice of vendor and support required from the vendor. In addition it should provide some guidance on acceptable risks, investment price, and limitations for the total cost of ownership.

- Manager's Philosophy

Every business has an internal structure designed to compliment the business process. In this model the manger is considered the decision maker. If he believes Information Technology is the key to success then he will push to upgrade the business software every time a new product is released to market. This may lead to a discontinuity between the user and organizational cycles. Referring to the user adaption cycle section; getting end-users to use a new system correctly may be much harder and more important than actually implementing the system itself because the success of a system is generally

measured on how well the system is used by end-users. There are many other factors that the manager must consider before making the decision to upgrade.

- Business Needs.

This is the most significant factor when considering internal requirements. This is where the balance between organizational functionality, business process and users' ability to adapt to software upgrades can be analyzed and possibly synchronized.

The External Dependencies consist of:

- Software Functionality

The possible benefit that the improved functionality of the new software offers plays a significant role in the decision process. However, when making this decision a manager must consider organizational resources, and the current organization's process. This is where software vendors try to influence organizations by reducing the time it takes to release newer versions of software packages and marketing the upgrades that they have to offer. Organizations do not have the time to train, adapt and see return on investment before they are supposed to upgrade again. This can cause discontinuity between the Organization Cycle and the Software Development Cycle because software vendors continue to improve their ability to produce software at a faster rate, resulting in an organization's inability to keep up.

- Technical Support

Vendors will continue to improve their software packages and release updated versions. This gives vendors a tremendous influence over the organizations upgrade decision cycle.

**b. Contingency Forces**

Contingency forces are not as defined in this model, however, they play a key role in the decision process. Internal resources play a significant role in an organizations decision to upgrade. The availability of internal resources may not be the deciding factor on the decision to upgrade, however, the lack of internal resources can be the deciding factor in the upgrade decision (Khoo, 2005). This implies that the motivating forces and internal resources act together to determine an organization's decision to upgrade.

**c. Decision**

The two arrows drawn from the Motivating Forces and Contingency Forces to the decision component indicates that the decision to upgrade is the outcome of the interaction between contingency forces and motivating forces. The arrow pointing from the decision component to impacts, indicates that the decision to upgrade will have a positive or a negative effect on the stake holders (Khoo 2005). In addition, this will effect the users adaptation cycle, the business process, and ultimately the business capital.

**d. Impacts**

The impact component will cause a positive outcome, negative outcome or a combination of both. Every decision to upgrade will incur cost, however, the hope is that this cost will provide a return on investment and additional benefits. Impacts are difficult to assess, however, a software upgrade can be assessed through the implementation process, changes in packaged software, and other circumstantial impacts. (Khoo 2005).

**e. Corrective Actions**

Corrective actions can be thought of as a coping strategy. This type of action is a reactive method devised to cope with the negative impacts that a software upgrade has already caused. An arrow drawn from impacts to

corrective actions indicates the reactive nature of this component. Additionally, a negative sign indicates that the corrective action is the response to the negative impact that the software upgrade has caused. An additional arrow is drawn from corrective actions back to impacts to show the changes that have been made to counter the problem. One last arrow has been drawn using a dotted line to indicate some of the formalization of the corrective actions that should be used as part of the planned strategy for the next upgrade (Khoo 2005).

#### ***f. Planned Strategies***

This strategy is for the sole purpose of reducing the negative impacts that a software upgrade may cause. It has several subcategories that play a role in mitigating negative software upgrade impacts. It has an arrow indicating a moderating relationship that is drawn from the planned strategy to the line connecting decisions and impacts (Khoo 2005).

These business needs and requirements are part of what influences the decision to upgrade. “It is the fact of the organization’s dependence on the environment that makes the external constraint and control of organizational behavior both possible and almost inevitable” (Pfeffer and Salanik, 1978).

## **2. Software Implementation**

As vendors become more efficient at developing, marketing and distributing their product, organizations need to become more efficient at deciding why to upgrade, knowing how to upgrade and knowing when to upgrade. Whether it is packaged software, Enterprise Resource Planning (ERP) software, Service as a Systems (SaaS) software, or another type, these three questions are important for the successful integration of an organizational upgrade. Burton Swanson and Ping Wang from the UCLA Anderson School of Management, conducted an analysis of 118 firms and identified that a business’s coordination is closely identified as a know-why factor, and management understanding and vendor support as a know-how factor. These factors were

identified as vitally important to the success of organizational implementation of ERP software. The following are some further explanations and findings that relate to this paper. Figure 6 summarizes the success model according to Swanson and Pang.

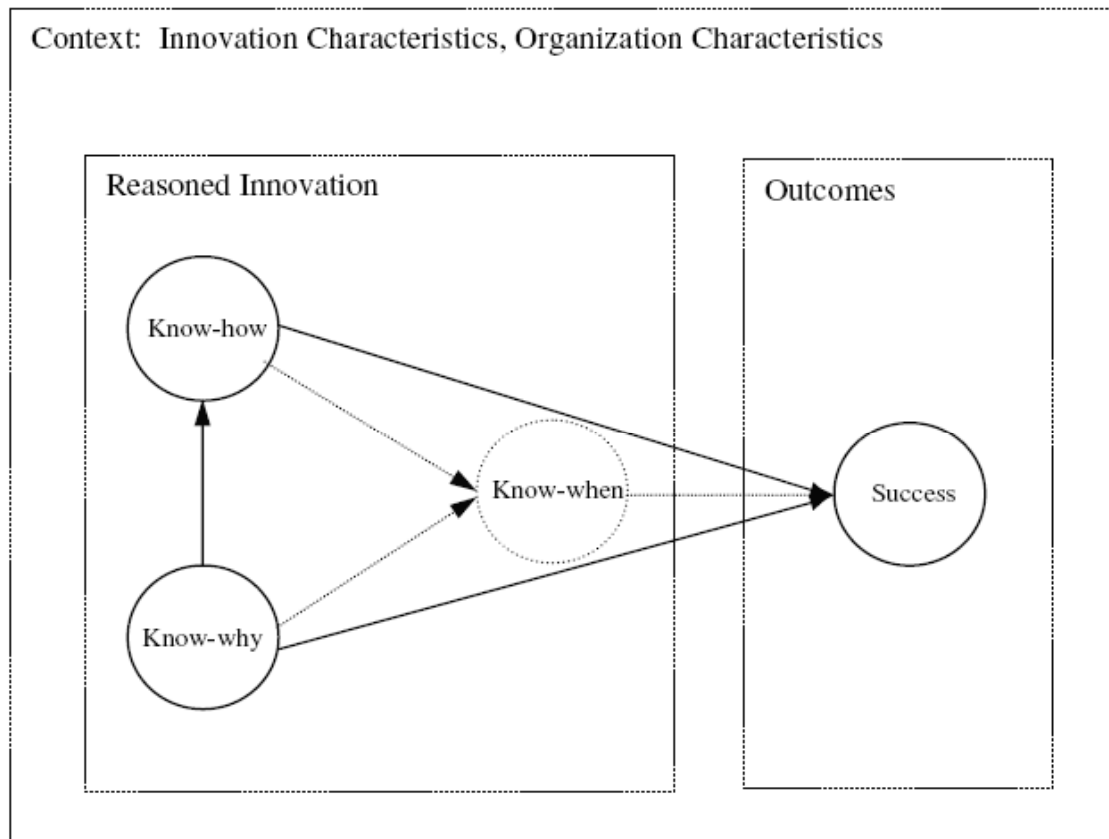


Figure 6. Reasoned Innovation Model  
From (Swanson and Wang 2005)

Specifications: (i) A firm's innovation reasoning will be composed principally of know-why (for adoption) and know-how (for implementation). Its know-how reasoning will be informed by its know-why. (ii) A firm's overall success with an innovation will depend substantially upon the know-why and know-how it brings to the undertaking. Assumptions: (i) Contextual variables will be reflected in a firm's innovation reasoning (and actions). (ii) A firm's know-when (for its actions) will be implied by its know-why and know-how (Swanson and Wang 2005).

Swanson and Pang collected their data using a multi-year mail in survey. They found that the 118 companies were of different sizes with a median of 5000 employees and quartile range of 1550-15000 employees. They also found that their Information Systems (IS) staff varied in a similar manner, with a median of 140 information specialists and the quartile range of 32-400 specialists. After a complete analysis of the responses collected by the 118 companies they developed some key factors that are relevant to Know-why, Know-how, and success. Then they listed several indicators for each of the factors. The indicators are the answers to some of the questions in the questionnaire. The three factors related to Know-why are characteristics that show an adoption rational. The four factors related to Know-how are characteristics that show an organizational software adaption structure. A description of these factors and their indicators are listed in Table 6 below.

<i>Factor</i>	<i>Item no.</i>	<i>Judgmental statement in questionnaire</i>
<i>Know-why</i>		
Business coordination	2	This package provides features which enable us to work better with our suppliers and/or customers.
	46	This package facilitates our user communications across departments.
Application integration	7	This package enables us to bridge easily to other applications in our portfolio.
	23	An advantage of this package is that it works with other packages from other vendors.
External authority	29	This package was initially evaluated and recommended to us by a consultant.
	48	Adoption of this package is a step toward outsourcing our IS department.
<i>Know-how</i>		
Vendor support	9	This package is well maintained by the vendor.
	14	This package is relatively free of bugs.
Project planning	6 <sup>a</sup>	We underestimated the amount of training needed to implement this package.
	13 <sup>a</sup>	We underestimated the amount of consulting we required to implement this package.
	25 <sup>a</sup>	We underestimated the time it would take us to implement this package.
Management understanding	35	Top management has provided us with the necessary resources to successfully implement this package.
	39	Our top management understands the costs to implement and maintain this package.
Custom requirements	4 <sup>a</sup>	We underestimated the amount of customization needed for this package.
	47 <sup>a</sup>	This package requires enhancements in order to better meet our needs.
<i>Success</i>	24	Most of our users of this package are on the whole happy with it.
	33	We are on track with our plans for the implementation and use of this package.
	55	On the whole, our implementation of this package has been successful.

Table 6. Key Factors  
From (Swanson and Wang 2005)

In order to explore the validity of their proposed model, they developed eight Structural Equation Models (SEM) using various combinations of the factors and their related indicators. The Models were labeled A-H. Swanson and Wang proposed that by comparing the relative fit for their multiple models they could show a logical progression using different factors on the same data. In addition, this would support specifications of theoretical models for future confirmatory studies. The models were constructed using a different combination of factors for the Know-why, Know-how, and success based on the limitations imposed by the questionnaire results and the 16 variable data shown in Table 6. "For instance, Model C consists of one Know-why factor (business coordination) and its indicators (Items 2 and 46), one Know-how factor (management understanding) and its indicators (Items 35 and 39), and the success factor and its indicators (Items 24, 33, and 55)" (Swanson and Wang 2005, p 24). After constructing the eight models they put each through a series of statistical tests. This mathematical analysis is outside the scope of this paper, however, the results are not. Table 7 summarizes the standardized estimation results and goodness-of-fit for these models. Note:  $R^2$  for success shows that Model C has the most variance (59%).  $R^2$  is the root-mean-square error of approximation (RMSEA), for more information on RMSEA, see Hu, L.-t. and Bentler, P.M. (1999). Cutoff Criteria for Fit Indexes in Covariance Structure Analysis: Conventional criteria versus new alternatives, *Structural Equation Modeling* 6(1): 1–55.

	Model								
	Item No.	A	B	C	D	E	F	G	H
<i>Know-how → Success</i>									
Vendor support	9, 14	0.47*				0.55*			
Project planning	6, 13, 25		0.11				0.34*		
Management understanding	35,39			0.37*				0.50*	
Custom requirements	4,47				0.21				0.38*
<i>Know-why → Success</i>									
Business coordination	2,46	0.51*	0.63*	0.59*	0.61*				
Application integration	7,23					0.18	0.24	0.29*	0.25*
<i>Know-why → Know-how</i>									
		NA	0.42*	0.24	0.32*	0.24*	0.22	0.05	0.15
<i>R<sup>2</sup> for success</i>		0.48	0.46	0.59	0.51	0.38	0.21	0.35	0.24
<i>Model parameters</i>									
Independence $\chi^2$		231.81	288.02	183.53	218.58	228.70	272.39	184.29	213.49
Degrees of freedom		21	28	21	21	21	28	21	21
Model $\chi^2$		28.18	41.45	14.65	15.16	10.84	34.76	19.97	15.41
Degrees of freedom		12	17	11	11	11	17	11	11
P-value		0.01	0.00	0.20	0.18	0.46	0.01	0.05	0.16
Bentler–Bonett non-normed fit index		0.87	0.85	0.96	0.96	1.00	0.87	0.90	0.96
Comparative fit index (CFI)		0.92	0.91	0.98	0.98	1.00	0.93	0.95	0.98
RMSEA		0.11	0.07	0.05	0.04	0.00	0.10	0.09	0.06
LISREL GFI		0.94	0.93	0.97	0.97	0.97	0.93	0.95	0.96
Number of iterations		7	7	7	7	6	7	7	7

\*Significant at 0.05 level.

Table 7. Models A-H  
From (Swanson and Wang 2005)

Swanson and Pang found that while all models proved to be beneficial in understanding software implementation, it was Model C that was best suited for explaining what factors are most important to implementation success. See Figure 7 for results on Model C. “Model C is our best model by both model-fit and  $R^2$  criteria. It is composed of the business coordination know-why factor and the management understanding know-how factor, both of which are significant in explaining implementation success” (Swanson and Pang 2005, p 27). In addition they found that the business coordination, vendor support, and management understanding factors play a significant role in the success of the software upgrade process.

These findings provide insight on some of the key factors necessary for an organization to be successful at software implementation.

### **3. Case Study on Windows Vista Implementation**

Dean Williams, a consultant for Softchoice Corporation conducted a study designed to determine if organizations were prepared to implement the software upgrade to Windows Vista (WV). The data from this study is an inventory collected from 112,113 desktops in 472 North American organizations. The organizations include companies from industry sectors in finance, healthcare, technology, education and manufacturing. The data was collected between June 1<sup>st</sup> and October 1<sup>st</sup> of 2006. Williams found that over half of today's organizations did not have the hardware needed to adopt the minimum requirements for WV, and over 94% of the organizations did not have the hardware needed to meet WV premium package. Within these groups, he found that the majority only required minor upgrades, such as upgrading storage space or RAM, however, the rest were in need of major upgrades, such as graphics cards, and new processors. The necessary upgrade results can be found in Figure 7 below. He attributed this to several factors:

- WV required a substantial increase in hardware resources
- Many organizations had a hardware lifecycle that was in excess of five years.
- A lack of easy access to the PC inventory information needed to implement an effective life cycle management.

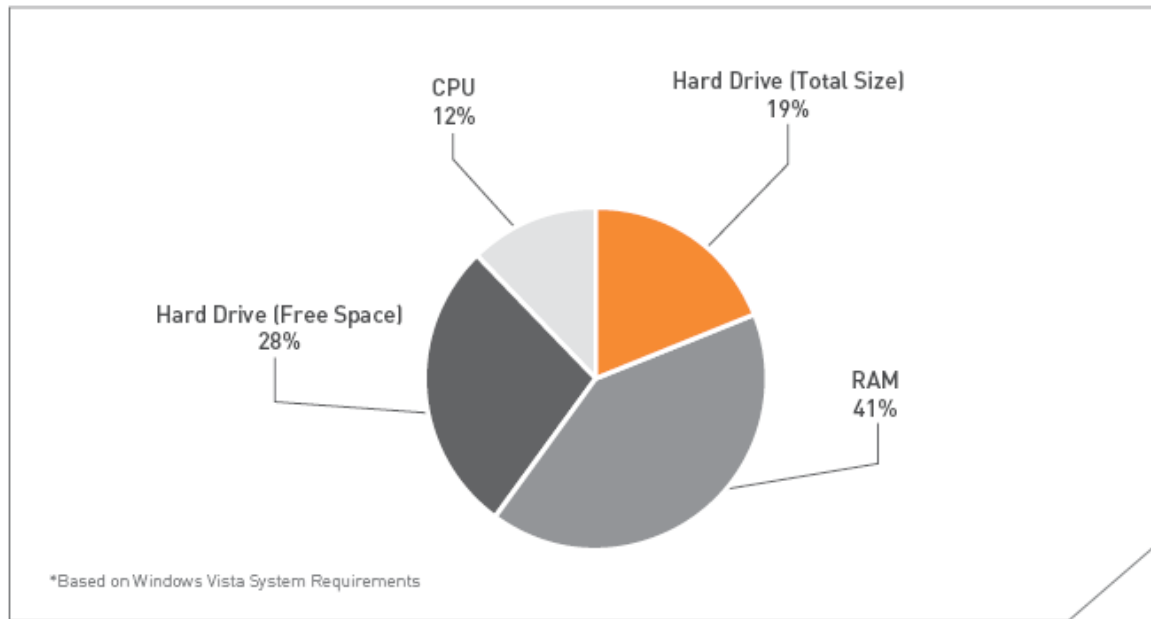


Figure 7. Components Requiring Upgrades within the 50% not meeting requirements. From (Williams 2006)

This study also indicated that from the release of Windows XP to the time of the study the PCs CPU speed had increased approximately 215 percent in five years, but the requirements for WV needed an increase of approximately 243 percent. Figure 8 shows a comparison in the system requirements needed for the last four Microsoft operating system releases.

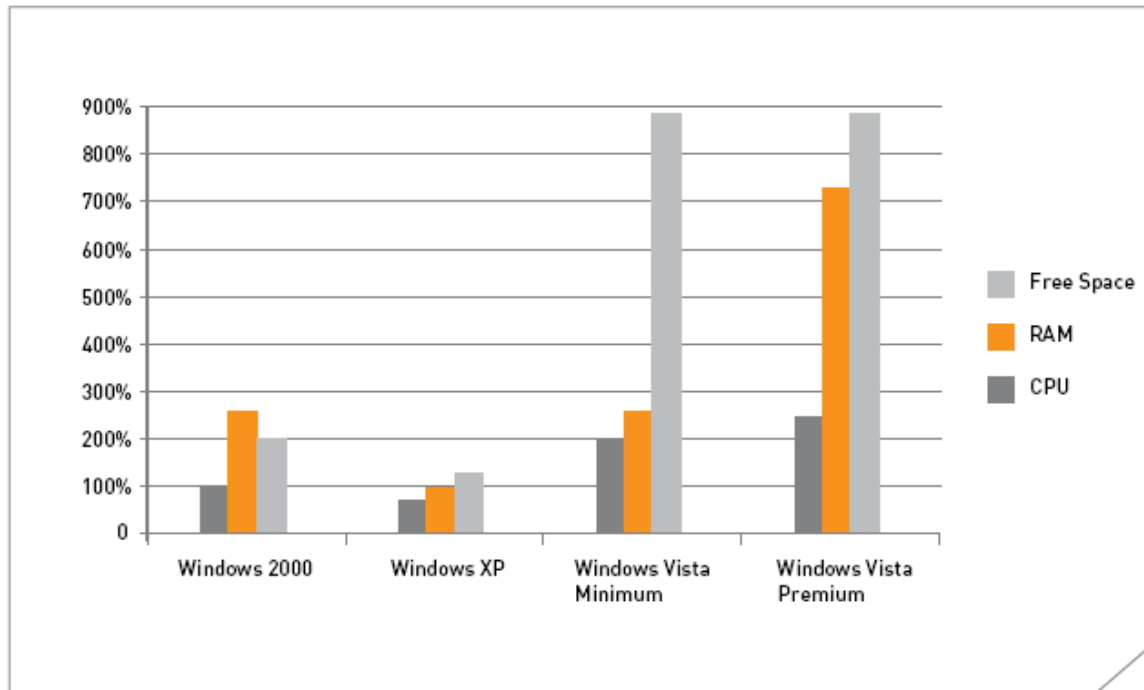


Figure 8. Percentage increases in system requirements, 2000-2006  
From (Microsoft)

**a. Future Preparedness**

The Williams study found that 27 percent of the organizations planned to wait one to two years before upgrading to WV. Another 33 percent planned to wait six months to one year before upgrading. This indicates that organizations do consider the software lifecycle. As organizations plan to upgrade to WV their PC lifecycle and current state is key to a smooth transition to WV. Swanson developed some recommendations which might be beneficial in planning the upgrade to Windows Vista. These recommendations can be seen on Table 8.

Vista Deployment Plan	Recommended Action	Benefits
Immediately	<ul style="list-style-type: none"> <li>• Assess the ability of the current environment to support Vista.</li> <li>• Determine which machines will require upgrading or replacing.</li> <li>• Forecast necessary purchases with chosen hardware vendor.</li> </ul>	<ul style="list-style-type: none"> <li>• Eliminate unexpected hardware expenses.</li> <li>• Maximize vendor discount potential from hardware forecasting.</li> <li>• Recoup optimal value from PCs slated for retirement.</li> <li>• Streamline upgrading and deployment process - Reduce unexpected IT resource strain.</li> <li>• Project total cost of Vista deployment.</li> </ul>
6 months – 1 Year	<ul style="list-style-type: none"> <li>• Assess which machines will not support Vista at the time of rollout.</li> <li>• Determine which machines will require upgrading or replacing.</li> <li>• Forecast necessary future purchases with chosen hardware vendor.</li> <li>• Adjust current hardware purchasing standards to meet Vista requirements.</li> </ul>	
1 Year – 2 Years	<ul style="list-style-type: none"> <li>• Assess which machines if any will not support Vista at time of rollout.</li> <li>• Adjust current hardware purchasing standards to meet Vista requirements.</li> </ul>	

Table 8. Actions and Benefits for Windows Vista Deployment  
From (Williams 2006)

While organizations tend to focus on upgrade implementation factors and lifecycle aspects of new software, the Williams case study shows that hardware lifecycle and purchasing decisions for that hardware are significantly related to a successful software upgrade.

### C. SOFTWARE INDUSTRY DEVELOPMENT CYCLES

Software vendors, both large and small, often produce and market their products faster than organizations and users can adapt to them. This trend is evident to software industry observers and practitioners alike. Reducing the time to market is a top priority for most software vendors as they try to take advantage of benefits such as extended market life, increased market share and greater freedom in pricing. Many companies try to gain competitive advantage by exploiting these benefits in the marketplace by shortening their development cycles and rushing their products to market, however, studies have shown that this practice can lead to undesirable consequences. The trade offs for pushing

software to market and reducing production cycle times leads to buggy software products, aberrant development practices and increased strain on software engineers (Sims 1997).

Table 8 below shows the evident shrinking cycles times according to a Product Development and Management Association (PDMA) survey of about 200 U.S. companies in a variety of industries.

Type of new product	1990 average development time	1995 average development time
Breakthrough products	49 months	42 months
New product lines	35 months	29 months
Major product revisions	22 months	18 months
Minor product revisions	10 months	8 months

Table 9. PDMA survey on shrinking cycles in product development  
From (Sims, D. 1997)

The shrinking cycle times displayed in Table 6 are from figures between 1990 and 1995, and reflect that decreasing time to market trends are not a new occurrence. Today, some observers claim that the Internet has added to the reduction of time to market methods because it has made it easier for vendors to distribute beta versions of products for testing. Although this theory is plausible and no studies were found to refute this theory, some suggest that it's not the proliferation in use of the Internet that has contributed to faster production development of software, but the market itself and today's global economy is to blame. Organizations and businesses worldwide are in a constant search for a competitive advantage and more often, management personnel are focusing on software and new technologies to expand the gap between their companies and the competition. Thus, reducing the time to market is more than just a goal for most vendors in the computer software industry; it has become the focus for survival in a highly competitive market.

## **1. Vendor Benefits from Improving Development Times**

There is no doubt that one of the most important considerations in commercial software development is that of meeting projected market release deadlines (Collier, K. and Collofello, J. 1995). The benefits reaped from improving software development times are many, but the most obvious benefit is the fact that the earlier the software is released the more time the product is given to generate sales and revenue. The introduction of software on the market sooner does not mean that it becomes obsolete any faster. Research in software cycle time development suggests that the time of obsolescence remains fixed regardless of the time of introduction. Additionally, studies indicate that switching software has a high cost because users of software tend to become loyal to their software products. This is mainly due to the high costs and learning times associated with switching to new applications. Another benefit lies in the fact that shorter development cycles can also increase market share. For example, the first product to market, which provides a new capability or new function, will initially hold 100% of the market share. This, together with the high switching costs, will increase a vendors ability to obtain and retain a large market share. Still, another benefit is the likelihood of higher profit margins. When a vendor speeds up development cycles, costs are likely to decrease. Additionally, shortening a products time to market allows for greater freedom in pricing. Earlier release of contract software usually means an increased chance in obtaining contracts earlier than their competitors and, together with lower development costs, may provide a greater profit margin. Last, shortening software cycle times allows engineers to start later in the development process, which allows them to take full advantage of any technological advances that constantly occur in today's fast paced technical environment.

## **2. Money, Time and Quality**

As eluded to in the paragraph above, software vendors have the opportunity to save money, increase profit margins and take advantage of ever

evolving technologies from increasing development cycle times, however, tradeoffs usually occur at the expense of quality and product performance. Software companies must make trade-offs to deliver products in a hurry. Money, time and quality are the main factors being juggled by vendors and eventually, one factor is relaxed at the expense of another. (Carmel 1995). In many cases today, developers may knowingly deliver buggy products in order to meet time to market and cost constraints. Recent studies suggest that time to market is the overriding concern for all managers and have shown that marketing managers are more concerned about features and platform diversity than about quality. For example, companies often inadequately document projects and inadequately staff quality assurance teams (Barr, A. and Tessler, S. 1996). Barr and Tessler also note that workers in quality assurance are often assigned a lower status than other members of project teams, giving them less influence over the development process and in some cases aren't even given enough time to test products before they go to market. Barr and Tessler's survey of eleven firms within the relational database management systems (RDBMS) and call center management segments of the software industry found that:

1. In both segments, (RDBMS and call center software), features, quality and cost were consistently traded off to achieve time to market.
2. No dominant software development methodology was identified. Only 50% of their respondents described any formal methodology. Most were developed in house.
3. A majority of teams allowed features to be added or dropped very late in the development process, even after beta testing.
4. Product decisions in more than half the firms were dominated by Engineering. Less than 20% of the respondents made decisions based on consensus between Engineering and Marketing.
5. Code re-use, from prior versions (91%), other company code (54%), and commercial products (27%) was very common.

6. Project level budgeting and cost tracking were low-priority activities. Head count was the only budget control mentioned.
7. Major differences in perception exist between the Marketing Product Manager and his or her counterpart in Engineering about such issues as how the requirements for the release were formulated, tradeoffs, the nature and source of development problems, etc.
8. Project teams were organized in a variety of ways. Functions like Quality Assurance and Release/Project Management could be located in several different departments and at different levels within the department, and even documentation and Tech Support held a variety of organization chart positions.

The results of the survey indicate that time to market is the key driver for a variety of software vendors, even at the expense of quality, managerial practice and employee considerations. Software industry executives often claim that the extremely rapid rate of change in their technologies and markets force them to make conscious tradeoffs among cost, quality and features in the course of their software development. Marketing managers, for example, conclude features and platform diversity to be the essential drivers contributing to competitive advantage. Consequently, tradeoffs between money, time and quality are having significant impacts on organizations and users of new software. As mentioned in section 1, a 2002 study commissioned by the National Institute of Standards and Technology found software bugs cost the U.S. economy about \$59.5 billion annually. The same study found that simply by improving testing through Quality Assurance could have mitigated more than a third of that cost –about \$22.2 billion.

Not only is money being potentially wasted because of the tradeoff situation, as described earlier in this chapter, user acceptance and resistance to adoption present further problems for organizational software upgrades as well.

These factors all contribute to the notion that disjointed cycles among users, organizations and the software industry are ever present in today's fast paced environment.

### **3. Lessons for the Software Industry**

As stated earlier, time to market is a fundamental competitive strategy in the software industry. It's safe to assume that today's software vendors face increasingly intense competition due to the proliferation of personal computer technology. However, software product development managers appear to be less concerned with, or even aware of, cycle time than they are with other competitive variables (Carmel, E. 1995). In a study conducted by Erran Carmel, 15 small to medium sized software package developers were found to be generally unaware of cycle time reduction as a management concept. He concluded that software development companies tend to focus on rapid development and deadlines and found that during peak periods of activity, 87% of the developers in core teams worked more than 56 hours per week and 47% worked more than 71 hours per week. These findings add to the speculation that developers are susceptible to becoming stressed-out and unhappier with shrinking cycle development time, which may end up having adverse effects on their products, such as quality and buggy software. This problem suggests that in order to remain competitive, software developers need to determine how they can better integrate quality assurance activities into their development process while reducing cycle time. In the survey he conducted on 15 software development firms, Carmel found that only 2 out of the 15 companies used process models or risk analysis techniques and that most could not articulate even an end-of-cycle tradeoff scheme that would represent an elementary type of risk analysis. Similarly, he found that they devoted scant resources to automated tools and relied heavily on software reuse and incremental innovation. Figure 8 below represents how much money the firms spent on automated tools. More than half the companies, for which there was data, spent less than 1000 dollars per year per developer on outside software tools.

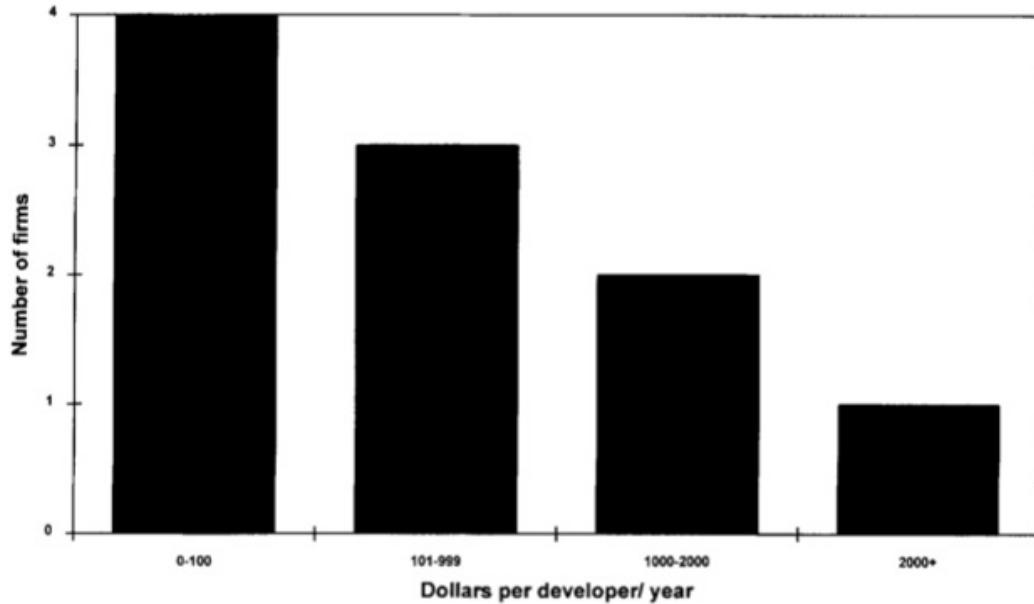


Figure 9. Spending on Software Design, Development and Testing Tools  
From (Carmel, E. 1995)

Although these companies reported that they made use of other sources to attain software tools, such as freeware, shareware, and trial versions of other vendors' software, the use of automated tools still appears low (Carmel, E. 1995).

In light of Carmel's findings on cycle time awareness, tradeoffs and cycle time reduction variables, he describes 5 lessons useful to software vendors in practice and theory. These 5 lessons are important to our study because they reveal vital steps software vendors can implement which can lead to reductions in cycle times and help users and organizations effectively and efficiently adopt new software.

#### **a. Lessons for Practice**

Lesson 1: The Core Team is a Key Success Factor. Carmel's study indicated that in almost all companies, the respondents pointed to team related factors as a key to their success and a key to their ability to develop software rapidly. Apparent was the fact that software companies consider a highly

effective, organic, entrepreneurial, organizational structure to be a key success factor in reducing cycle times. For example, if a company's cycle time was considered to be "good" (fast, approaching optimum), it was mostly contributable to the organizational variables listed above. If their cycle time was considered to be "inadequate", the companies attributed it to the lack of organizational management variables. The sample firms surveyed in the Carmel study encompassed firms that were, by and large, young companies where their organizational team structures formed organically. A well functioning organic team is the strongest type of team (Hofstrand, D. 2007). It displays synergy - the results produced by the team are greater than the sum produced by each individual team member. Synergy occurs because the team uses the strengths of each member while minimizing the weaknesses of each. The organic team is often used in situations where all of the team members know how to carry out the responsibilities of each of the team members, but inherently important in this structure is the need to have an intimate knowledge of the other team members. Our findings suggest it is quite apparent that within larger software firms, organic team structures need to be fostered and nourished to replicate the desirable entrepreneurial spirit usually found in smaller software development teams.

Lesson 2: Quality Issues Significantly Affect Cycle Time and Merit More Managerial Attention. Carmel notes that the software product category has inherent quality problems and he argues that there is no other product category in which products are routinely released with so many defects. Testing is identified as one of the largest components in quality assurance and is also one of the most difficult tasks in software development category. This is because any software program can have an infinite number of logic branches and testing every logic branch is too monumental of a task, taking up valuable time and resources. Although no evidence suggests that quality assurance activities shorten or lengthen cycle time, it is evident that a quality control regime involved in the process implies that if things are done right early, vendors can decrease expensive error correction late in the development cycle. Today, software packages are released with long lists of known defects and bugs, however, it is

still not clear in today's market, whether the market would wait for near zero defect software products. Zero defect products may also be rejected by the market because of the high cost associated with high quality assurance standards. Thus, appropriate risk analysis techniques need to be developed and studied to manage these tradeoffs correctly. Carmel suggests that as the software industry matures and users become more demanding with regards to quality, more attention needs to be given to quality assurance and near defect-free software. Further studies in this field are needed to identify the effects this will have on time to market, but as of now, observers in this field agree that unless active steps are taken to integrate quality assurance into the development process, near zero-defect strategies will have an adverse impact on a product's time to market.

### ***b. Lessons for Cycle Time Theory***

Lesson 3: Trade-off Research Needs to Include Quality. Carmel describes the tradeoff concept as one in which "units" of quality are decreased to allow additional "units" of features within a given time period. Table 9 shows 5 of the 15 firms' responses to the statement: Given 100 possible total points, allocate the points to the 5 items, weighted by importance to success. The 5 items being Cycle Time, Features, Quality/Defects, Cost of Development, and Product Pricing.

Firm	Points Allocated to .....					Total
	Cycle Time	Features	Quality/ Defects	Cost of Development	Product Pricing	
3	50	30	10	10	N/A	100
4	50	10	20	20	N/A	100
5	40	40	20	0	N/A	100
11	0	40	25	35	0	100
14	15	80	3	1	1	100
Average	31	40	16	13	N/S	100

Figure 10. Self-Reported Trade Offs for Successful Product Development  
From (Carmel, E. 1995)

The figure shows how software development companies view the items cycle time and features as much more important to overall product success than quality and defects. Tradeoff results depicted in Figure 6 and earlier discussions on software defects show that software managers trade units of quality for other product goals whether explicitly or implicitly. Further research in this field needs to be conducted but it is important to note that tradeoffs in product innovation can be understood within the framework of risk analysis techniques. Understanding the principals and practices of risk analysis in software development can help developers understand the consequences of tradeoffs more thoroughly. Additional research on cycle times should focus on the examination of the relationships between managers' tradeoff preferences, the deliberate actions they take and cycle time outcomes.

Lesson 4: Distinctions Between Cycle Time Awareness Levels Affect Cycle Time Reduction Behavior. Carmel found that software development firms' actions and processes are driven by what is termed as a desire for rapid development, rather than deliberate action driven by overall cycle time considerations. This distinction was made to differentiate pre-cycle and early-cycle developmental approaches from reactive, late-cycle approaches. Because cycle time research is concerned with strategies for cycle time reduction rather than other coincidental factors, it must examine the awareness of cycle time reduction as various strategies and actions are applied. This methodology is necessary to help facilitate understanding which strategies work and which ones just happen to be present. That is to say, we need to understand which specific actions managers take to reduce cycle time, rather than to achieve other goals such as better design, lower cost, etc. This in-depth kind of research will allow for greater understanding of the variables that have a true impact on cycle time reduction.

Lesson 5: Software Versions Are Tied to Market Rhythms that May Lessen Pressures for Cycle Time Reduction. Software package versions create both positive and negative effects on cycle time reduction. The positive effect is that software versions are typically advanced in incremental innovation. New

versions encompass incremental changes thus contributing more easily to quicker cycle times. The negative effect in the version concept is manifested through an entrapment into the software market rhythm. Most of the software industry releases new version every 12 to 24 months. This is viewed by the bulk of personal computer software users to be common managerial strategy in this industry. Carmel notes that the structural dynamics of the market rhythm are such that customers have an ambivalent approach to adopting new versions. On one hand, they desire the latest features offered by the new version, but on the other hand, they realize that the upgrade involves at least some investment of time or other resources like training budgets. For this reason, few customers choose to upgrade to new versions too frequently. Consequently, both customers and vendors tend to slip into the market rhythm, which cycles at 12 to 24 months. This phenomenon requires future studies relating industry specific market rhythms when examining and comparing cycle times. Establishing bench mark cycle times within various software companies and organizations will shed further light on which improvements can be compared.

THIS PAGE INTENTIONALLY LEFT BLANK

## **IV. CONCLUSIONS AND RECOMMENDATIONS**

### **A. ANALYSIS AND RECOMMENDATIONS**

The software industry is in the midst of a dramatic revolution according to 2006 software industry report conducted by McKinsey & Co. and the Sand Hill Group. The implications of this revolution are increased innovation, new business models, technology discontinuities, and global capability shifts. The report indicates that the software industry is experiencing an increased flow of internal and external capital and substantial innovation as well as increased private equity investments and expected increases towards IT budgets in the coming years. In fact, software budget growth is expected to gain 5% between 2006 and 2008, from 30% to 35% (Berryman et al. 2006). Taken together, these factors and changes will have profound implications for software providers, organizations and users.

This chapter will focus on identifying possible recommendations for software executives, managers and leaders where inevitable cyclic discontinuities exist between their organizations, users and the software industry. In taking into consideration the various advantages and disadvantages within user cycles, organization cycles, and the software industry's cycles, we intend to shed further light on what IT managers can learn, expect and take action on.

First, it is important to list and describe those attributes, which are considered to be advantages and disadvantages with respect to software cycles. In Figure 2 of Chapter II, we introduce the conceptual framework of the Software Upgrade Cycle. Below is an extension of the same figure with corresponding driving forces associated with the cycles of users, organizations and the software industry.

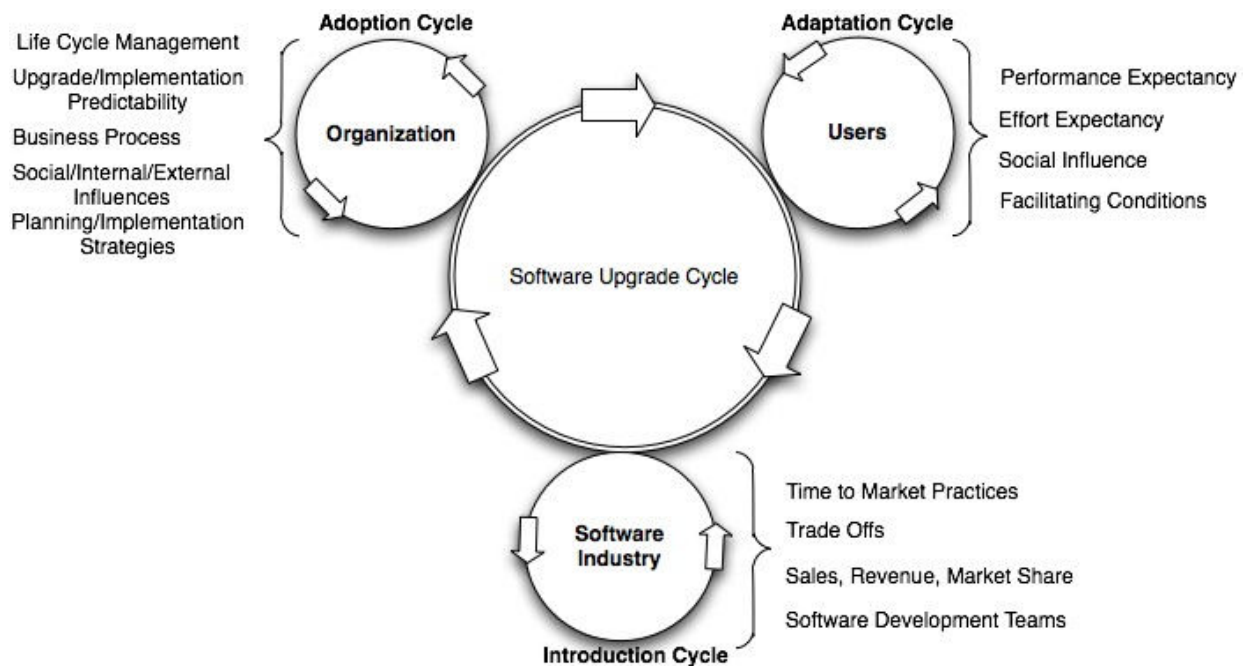


Figure 11. Software Upgrade Cycle with Corresponding Driving Forces

As described in the preceding chapters, driving forces within organizational adoption cycles, user adaptation cycles and introduction cycles within the software industry influence the software upgrade cycle phenomenon. To better delineate these moderators, which effect each cycle, we introduce the advantages and disadvantages inherent among the driving forces. The tables presented on the following pages describe attributes which positively and negatively effect the software adoption cycle as a whole and therefore should be of vital importance from the perspective of executives, managers and leaders within the IT industry. The first table delineates advantages and disadvantages of the user adaptation cycle.

<b>Advantages and Disadvantages of the User Adaptation Cycle</b>		
	<b>Advantage</b>	<b>Disadvantage</b>
<b>Performance Expectancy</b>	Men and younger women believe use of the system may allow increased productivity, effectiveness, ease of use and faster accomplishment rates. Rewards and benefits from using the system may be perceived to be higher among this group.	Job related factors become less significant for working women between the time they enter the labor force and the time they reach child-rearing years. This group may be less inclined to initially perceive the rewards and benefits of a new system as advantageous.
<b>Effort Expectancy</b>	Women, particularly younger women in the early stages of experience may perceive use of the system to be free of effort, easy to understand and easy to use or interact with. Women may initially be more accepting of the new system as a result.	Men and older women may be less inclined to initially accept the change to a new system. Perceptions may lead them to believe the degree of ease and flexibility associated with learning the system may be difficult.
<b>Social Influences</b>	Women, especially older women and older workers in general are more sensitive to others' opinions and place increased salience on social influence. This group may be more voluntary in acceptance and initial use of a new system.	Younger workers and workers in later stages of experience may place less salience on social pressures and influences. This group may be less inclined to accept use of a system based on others' expectations even when those referent others have the ability to reward the desired behavior or punish non-behavior
<b>Facilitating Conditions</b>	Older workers, particularly those with more experience attach more importance to receiving help and assistance. This group perceives given the resources, guidance, opportunities and knowledge it takes to use the system, the system would be easy to use and compatible with aspects of their work.	Younger workers with less experience may place less importance on receiving help and assistance with a new system within their organization.

Table 10. Advantages and Disadvantages of the User Adaptation Cycle

The second table delineates advantages and disadvantages within the software industry's introduction cycle.

<b>Advantages and Disadvantages within the Organization Adoption Cycle</b>		
	<b>Advantages</b>	<b>Disadvantages</b>
<b>Life Cycle Management</b>	<ul style="list-style-type: none"> <li>- Replaces old outdated legacy systems</li> <li>- Improves user interface and increase user satisfaction</li> <li>May increase productivity</li> </ul>	<ul style="list-style-type: none"> <li>- Compatibility complications with legacy software or systems often cause projects to go over budget and exceed expected project completion dates.</li> <li>- May decrease productivity in initial stages of implementation</li> </ul>
<b>Upgrade/Implementation Predictability</b>	<ul style="list-style-type: none"> <li>- Improved cost analysis</li> <li>- Improved functionality</li> <li>- Improved perception of reliability</li> </ul>	<ul style="list-style-type: none"> <li>- Adoption of poorly tested software or poorly QA tested software requires rework, bug fixes and patches or further upgrades</li> <li>- Unpredictable software often leads to dissatisfied user experiences</li> </ul>
<b>Business Process</b>	<ul style="list-style-type: none"> <li>- Standardization of business process</li> <li>- May help to implement change and enhance competitive edge</li> </ul>	<ul style="list-style-type: none"> <li>- Become incompatible with some customers who still have legacy systems.</li> </ul>
<b>Social, Internal and External Influences</b>	<ul style="list-style-type: none"> <li>- Improve customer relations and customer service</li> <li>- Improve employee job satisfaction</li> <li>- Portray a successful business image with the most up to date software.</li> </ul>	<ul style="list-style-type: none"> <li>- Bad press or negative public perceptions about an organization's IT systems may effect stocks, sales</li> <li>- Pressures to contribute more assets to IT may take away assets from other aspects of an organization</li> </ul>
<b>Planning/Implementation Strategies</b>	<ul style="list-style-type: none"> <li>- Facilitates improved testing</li> <li>- Establishes good foundations in IT policy</li> <li>- Provided experience in implementation approaches</li> <li>- Enhances learning</li> <li>- Helps to mitigate negative impacts caused by software upgrades</li> </ul>	<ul style="list-style-type: none"> <li>- Takes time</li> <li>- Untested strategies may require rework in production phases</li> <li>- Lends to tradeoffs if project deadlines can't be achieved.</li> </ul>

Table 11. Advantages and Disadvantages within the Organization Adoption Cycle

The third table delineates advantages and disadvantages within the software industry's introduction cycle.

<b>Advantages and Disadvantages within the Software Industry's Introduction Cycle</b>		
	<b>Advantages</b>	<b>Disadvantages</b>
Time to Market Practices	<ul style="list-style-type: none"> <li>- Get products to market before competitors</li> <li>- Release a greater amount of products in the same amount of time</li> <li>- Avoid criticism in the media about not having up-to-date products</li> <li>- Deliver products before customers can change their requirements</li> </ul>	<ul style="list-style-type: none"> <li>- Buggy products</li> <li>- Inadequate product documentation</li> <li>- Reliance on aberrant development practices</li> <li>- Increased strain on engineers</li> <li>- Customer aggravation about systems changing before they can be fully deployed</li> </ul>
Trade Offs	<ul style="list-style-type: none"> <li>- Cycle time reduction</li> <li>- Cost of development may decrease</li> </ul>	<ul style="list-style-type: none"> <li>- Dropped features</li> <li>- Quality decreases and defects increase</li> <li>- Product support may decrease</li> <li>- Lends to increased patches and upgrades later in cycle</li> </ul>
Sales, Revenue, Market Share Strategies	<ul style="list-style-type: none"> <li>- Allow product pricing to be flexible</li> <li>- Tested strategies may support competitive advantage</li> <li>- Saving money on development allows for larger budgets in other business areas such as marketing</li> </ul>	<p>Decreased budgets, time constraints adds pressure to development teams and lends to tradeoffs</p> <p>Lends to increased product revision cycles</p> <p>Untested strategies may decrease competitive advantage</p>
Development Teams	<ul style="list-style-type: none"> <li>- Cross functional team structure leads to better cooperation, faster decisions, shared responsibilities</li> <li>- Small teams have positive effect on intra-team communication</li> <li>- Success is determined by application specific experience of team members</li> </ul>	<ul style="list-style-type: none"> <li>- Dispersed or non co-located teams have difficulties communicating</li> <li>- Inexperienced team members may cost in time, money and rework required</li> </ul>

Table 12. Advantages and Disadvantages within the Software Industry's Introduction Cycle

## **B. TIMING AND FLEXIBILITY**

As this study suggests, many factors effect the timing associated with users, organizations and the software industry such as software life-cycle trends, environmental factors, market rhythm, budgets, management practices, policies and models, just to name a few. Most software companies release new major versions of their products every 12 to 24 months and minor revisions every 6 to 12 months, while upgrades and service patches can be introduced in a matter of days (Carmel, E. 1995 and Sims, D. 1997). Although more research needs to be completed in user acceptance timing, the Venkatesh et al. and Seymour et al. studies indicate that the majority of users, despite usage intent fluctuations, can adapt to major revisions or completely new versions of a single software application within 12 months. More complex software packages including Software as a Service (SaaS) packages and business solutions software packages, such as those products produced by SAP, PeopleSoft, IBM and Oracle, may take longer than 12 months for an organization's users to adapt to successfully. Integrating these more complex solutions or multiple software packages simultaneously can present organizations with myriad obstacles in the areas of user adaptation and organizational implementation. From an executive or managerial perspective, these obstacles need to be understood, anticipated and managed appropriately. The three tables we have presented in this section are relevant in that they identify both moderators and advantages and disadvantages managers can use towards a software integration plan. Both large and small organizations alike should be able to correlate and use at least some, if not all, of the aspects outlined in this section's tables in IT implementation processes. Admittedly, every organization will be different in size, managerial dynamics, policy processes, employee composition and other factors. But in some way, every organization will use some type of technology infrastructure and it is for this reason, the following tables will prove beneficial. As organizations decide to upgrade and integrate technological aspects within their various processes, our suggestions represented in the tables will allow management to focus on how their users will adjust to the adaptation and implementation

strategies employed. Our suggestions and recommendations concerning the various software vendors they rely on will give managers a better idea about when implementing new software might be appropriate and advantageous in terms of competitive advantage perspectives. Finally, our suggestions and recommendations presented in the organization table will provide managers with reasonable explanations as to why their organization should consider new software implementations and integrations.

### **C. STARTING POINT IN USING THIS RESEARCH METHOD**

Inevitably, executives and managers will ask these few and seemingly simple questions, which permeate not only the IS field but the manufacturing, marketing, sales, supply and human resource fields as well: What do I want/need, what can I afford and what or how can I integrate quickly. This section will focus on these questions as a starting basis while applying the software upgrade cycle and corresponding driving forces presented in this paper, as a tool to prepare practitioners and managers for a software upgrade. The following tables will outline the moderators and driving forces of each cycle (software industry, user, organization) and then present proposed considerations and actions to be explored. The proposed solutions are not all encompassing as there are too many influencing factors which may affect each particular upgrade; however, these are common considerations and actions that should be thoroughly explored when deciding to implement major and minor software upgrades. First, we introduce those considerations and actions to be explored from the software industry perspective.

<b>Software Industry Proposed Considerations and Actions</b>	
Time to Market Practices	<ul style="list-style-type: none"> <li>- Explore the reasons why a software manufacturer or vendor has released their product and under what circumstances. When was their last major revision or upgrade and how often do they introduce patches or revisions? These questions relate to the software company's business practices and strategies.</li> <li>- Explore, if feasible, what trade-offs were made prior to release. Explore the company's development process and team dynamics (size, pressures, incentives, experience).</li> <li>- Try to ascertain the company's incentives in product release timing and make any correlation to sales and market share strategies. Are these correlations consistent with their prior major/minor releases?</li> <li>- Assess the company's reliability as it pertains to Quality Assurance and product support.</li> <li>- Explore any other organization's prior implementation of the software and make note of important lessons learned or compatibility issues.</li> </ul>
Trade-offs	
Sales, Revenue, Market Share Strategies	
Development Teams	

Table 13. Software Industry's Cycle and Proposed Considerations and Actions

The next table presents those considerations and actions, which should be explored from the organization perspective. Again, these considerations do not encompass all aspects that should be studied, but merely represent a subset of common hurdles and influencing factors of which executives and managers should be fully aware of.

<b>Organization Proposed Considerations and Actions</b>	
Life Cycle Management	<ul style="list-style-type: none"> <li>- Assess where your organization is with respect to the life cycle of your present software systems and determine if and what kind of upgrades are feasible (major revision, minor revisions, patches) Are those revisions compatible with the current architecture?</li> <li>- Assess how predictable the proposed upgrade is with respect to Quality Assurance and testing initiatives conducted by the vendor/manufacturer?</li> <li>- Explore your organization's business and implementation processes and policies. Does your organization have a codified process and how effective was it in previous implementations?</li> <li>- Become familiar with the reasons why your organization is considering upgrading and determine if influencing factors align with current business goals. Is it feasible according to cost, time and benefits gained?</li> <li>- Assess your organizations capacity to effectively plan and implement new software. Is there a current strategy your organization uses or can adopt from elsewhere?</li> </ul>
Upgrade/Implementation Predictability	
Business Process	
Social/Internal/External Influences	
Planning/Implementation Strategies	

Table 14. Organization Cycle and Proposed Considerations and Actions

The last table presents those considerations and actions, which should be explored from the user perspective. Each organization's users can represent a unique mixture of experience, gender and age and also may comprise as many as a hundred thousand to as little as a dozen users. This diversity can lead to conflicting adaptation consequences, nevertheless, each executive or manager must decide on appropriate implementation strategies, adapt to changes and mitigate problems as best they can. The below recommendation should prove valuable to a great many managers as they consider the composition of users within their organization.

<b>User Proposed Considerations and Actions</b>	
Performance Expectancy	<ul style="list-style-type: none"> <li>- Determine the experience, age and gender composition of your organization. Explore the factors of performance expectancy and effort expectancy as they relate to specific gender and age attributes. Does your organization comprise a majority of men or women and how does this effect how they interpret ease of use and effort required?</li> <li>- Determine the amount of users who may be more susceptible to social influence and develop training and implementation strategies that work in unison with this factor (i.e. Mentor programs)</li> <li>- Assess the conditions that may improve or place more importance on receiving help and assistance from others in the organization. Take steps towards improving the facilitating conditions that may be weak in areas (i.e. encourage younger workers to ask questions and seek help from more experienced workers.</li> </ul>
Effort Expectancy	
Social Influence	
Facilitating Conditions	

Table 15. User Cycle and Proposed Considerations and Actions

#### **D. FUTURE DIRECTIONS CONCERNING THE USER CYCLE**

Venkatesh et al. concluded that the Unified Theory of Acceptance and Use of Technology (UTAUT) theory explains as much as 70 percent of the variance in intention to use new software. It is possible, however, that we could very well be approaching the practical limits of our ability to explain individual acceptance and usage decisions in organizations. Future research therefore should focus on identifying additional constructs and moderators that can add to the prediction of intention and use behavior over and above what is already understood. The theories used in this paper facilitate the advancement of individual acceptance research by unifying several theoretical perspectives presented by Venkatesh et al. and Seymour et al. Common in this literature is the incorporation of four moderators to account for the dynamic influences including organizational contexts, user experience and demographic characteristics. Further research focusing on these and other factors can shed further light on

this phenomenon and add to the proliferation of competing explanatory models, which intend to explain individual acceptance of information technology.

#### **E. FUTURE DIRECTIONS CONCERNING THE SOFTWARE INDUSTRY CYCLE**

It is apparent that cycle time reduction will continue to be a primary goal of competitive software development organizations for the foreseeable future, yet it is questionable whether or not a significant reduction in cycle time can be achieved at a low cost, with minimal impact on software quality and performance (Collier and Collofello 1995). It is therefore imperative for practitioners to gain a better understanding of what leads the software industry to the reduction of cycle time. A number of future research directions should be focused in the direction of trade off factors. This important aspect should be further explored, especially with respect to the role of quality assurance as well as the covariance of cycle time awareness with cycle time outcomes. In general, the software industry needs to explore ways to increase quality assurance objectives while decreasing cycle time. Other industry efforts need focus in establishing cycle time benchmarks, software reuse strategies and other organizational and use of experience techniques.

#### **F. FUTURE DIRECTIONS CONCERNING THE ORGANIZATION CYCLE**

When innovating with IT, organizations have arguably given less attention to the lasting quality of their strategic know-why reasoning discussed in chapter 3, than they have to the metrics of the investment moment (Swanson and Wang 2005). Comparative organizational case studies and innovation case studies focused on how and why organizations adopt new technologies are required to further understand their strategic reasons. Our findings suggest that further studies should examine in depth how organizations go about comprehending, adopting, implementing and assimilating an IT innovation. Swanson and Wang recommend that studies in this field should focus on the know-how as it effects implementation and assimilation and the know-why as it effects adoption, as well

as the interplay between both forms of knowledge. Other case studies might examine how this know-why and know-how contrast according to both organizations and their various innovations.

## LIST OF REFERENCES

- Barr, A. and Tessler, S. (1997) A Pilot Survey of Software Product Management. Stanford Computer Industry Project: SCIP Software Industry Study.
- Bem, D.J., and Allen, A. (1974) On Predicting Some of the People Some of the Time: The Search for Cross-Situational Consistencies in behavior. *Psychological Review* (81:6) pp. 506-520.
- Berryman, K., Jones, J., Manyika, J., Rangaswani, M. (2006) Software 2006 Industry Report conducted by McKinsey & Company, Inc. and the Sand Hill Group. Resource available at: [http://www.sandhill.com/conferences/sw2006\\_materials/SW2006\\_Industry\\_Report.pdf](http://www.sandhill.com/conferences/sw2006_materials/SW2006_Industry_Report.pdf). Last accessed June 2008
- Brown, S.A., Massey, A.P., Montoya-Weiss, M.M. and Burkman, J.R., (2002). Do I really have to? User Acceptance of Mandated Technology. *European Journal of Information Systems*, (11:4) pp. 283-295.
- Carmel, E. (1995). Cycle Time in Packaged Software Firms. *Journal of Product Innovation Management*. (12) pp. 110-123.
- Collier, K. and Collofello, J. (1995) Issues in Software Cycle Time Reduction. Conference Proceedings of the 1995 IEEE Fourteenth Annual International Phoenix Conference on Computers and Communications. pp. 302-309.
- Hofstrand, D. (2007) Designing Family Business Teams. *Ag Decision Maker*. C4-73. <http://www.extension.iastate.edu/AgDM/wholefarm/pdf/c4-73.pdf>. Last accessed June 2008
- Hall, D., and Mansfield, R. (1995) Relationships of Age and Seniority with Career Variables of Engineers and Scientists. *Journal of Applied Psychology* (60:2) pp. 201-210.
- Jasperson, J., Carter, P.E. and Zmud, R.W. (2005). Conceptualization of Post-Adoptive Behaviors Associated with Information Technology Enabled Work Systems. *MIS Quarterly* (29:3) pp. 525-567.
- Khoo, M., Huoi (2005). Up. Diss. Upgrading Packaged Software: An Exploratory Study of Decisions, Impacts, and Coping Strategies from the Perspectives of Stakeholders, 2005. Atlanta: Georgia State University, 2005.
- Lucas, H. C., E. J. Walton, et al. (1988). "Implementing Packaged Software." *MIS Quarterly* 12(4): 537-549.

- Minton, H.L., and Schnieder, F.W. (1980) Differential Psychology. Waveland Press, Prospect Heights, IL.
- Moore, G. (1965) Cramming more Components Onto Integrated Circuits. Electronics, Volume 38, Number 8.
- Nah, F.F., Tan, X., and Teh, S.H., (2004) An Empirical Investigation on End-Users' Acceptance of Enterprise Systems. Information Resource Management Journal, (17:3) pp. 32-53.
- Oliver, D., Whymark, G., and Rohm, C. (2005) Researching ERP Adoption: An Internet-Based Grounded Theory Approach. Online Information Resource Management Journal (17:3) pp. 32-53.
- Paine, M. (2000). Making Software Upgrades A First-Class Experience. Health Management. 21: 22-23.
- Pfeffer, J. and G. R. Salancik (1978). The External Control of Organizations A Resource Dependence Perspective. New York, Harper & Row, Publishers.
- Platt, D. (2007). Why Software Sucks. Boston, MA: Pearson Education Inc. 2-3
- Plude, D., and Hoyer, W. (1985) Attention and Performance: Identifying and Localizing Age Deficits. Aging and Human Performance, N. Charnes (ed.), John Wiley & Sons, New York. Pp. 47-99.
- Porter, M. (1985). Technology and Competitive Advantage. Journal of Business Strategy (5:3) pp. 60.
- Seymour, L., Makanya, W., Berrange, S. (2007) End-Users' Acceptance of Enterprise Resource Planning Systems: An Investigation of Antecedents. Proceedings of the 6<sup>th</sup> Annual ISOnEworld Conference, April 11-13, 2007, Las Vegas, NV. Resource can be found at: <http://www.information-quarterly.org/ISOWProc/2007ISOWCD/PDFs/26.pdf>. Last accessed June 2008
- Sims, D (1997) Vendors Struggle with Costs, Benefits of Shrinking Cycle Times. Computer 1997 Annual Index. (30:12) pp. 70-84.
- Swanson, B. and Wang, P. (2005). Knowing why and how to innovate with packaged business software: Journal of Information Technology. 20, 20-31

- Venkatesh, V., Morris, M., Davis, G., Davis, F. (2003). User Acceptance of Information Technology: Toward A Unified View. *MIS Quarterly* (27:3) pp. 425-478.
- Venkatesh, V., and Morris, M.G. (2000) Why Don't Men Ever Stop to Ask For Direction? Gender, Social Influence, and Their Role in Technology Acceptance and Usage Behavior. *MIS Quarterly* (24:1) pp. 115-139.
- Warshaw, P.R. (1980) A New Model for Predicting Behavioral Intentions: An Alternative to Fishbein. *Journal of Marketing Research* (17:2) pp. 153-172.
- Williams, D. (2006). Lack of Vista Readiness Pushes PC Lifecycle Management to the Forefront. Softchoice Corporation.

THIS PAGE INTENTIONALLY LEFT BLANK

## INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center  
Ft. Belvoir, Virginia
2. Dudley Knox Library  
Naval Postgraduate School  
Monterey, California
3. Mr. Glenn Cook  
Naval Post Graduate School  
Monterey, California
4. Dr. Thomas Housel  
Naval Post Graduate School  
Monterey, California